НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр Коваль
«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні технології моніторингу довкілля»

спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Проектування та розроблення WEB – додатків на платформі систем контролювання доступу "Intteks Acs"»

Виконав (-ла):
студент (-ка) IV курсу, групи ТМ-61
Кочкарьов Сергій Вадимович _____

Керівник:
Професор
Сігайов Андрій Олександрович _____

Рецензент:
старший викладач кафедри АЕС і ІТФ, к.т.н.
Воробйов Микита Валерійович

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____2

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 122 Комп’ютерні науки та інформаційні технології

Спеціалізація Інформаційні технології моніторингу довкілля

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту
Кочкарьову Сергію Вадимовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Проектування та розроблення WEB – додатків на платформі систем контролювання доступу “Intteks Acs”

керівник роботи Сігайов А. О. Доктор економічних наук
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020р. № 1168-с

2. Строк подання студентом роботи 10.06. 19

3. Вихідні дані до роботи React, Javascript, MongoDB, Android

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)Проаналізувати предметну область охоронних систем та розробити систему контролю доступу, що складається з : веб-сервер, мобільний додаток, панель адміністратора. Протестувати систему та проаналізувати результати.

5. Перелік ілюстративного матеріалу
"Аналіз сучасних тенденцій", "Огляд існуючих рішень", "Різниця між UML та діаграмами системи", "Засоби розробки системи", "Середовище розробки Vscode", "Мова програмування JavaScript", "React Native", "Clockify", "Опис програмної реалізації системи", "Реалізація серверної частини","Реалізація мобільного додатку. Мобільний зчитувач карт-пропусків", "Реалізація панелі адміністратора", "Взаємодія з мобільним додатком", "Взаємодія з панеллю адміністратора"

6. Дата видачі завдання «30 Вересня 2019р».

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	30.09.2019	
2.	Вивчення та аналіз задачі	25.11.2019	
3.	Розробка архітектури та загальної структури системи	6.01.2020	
4.	Розробка структур окремих підсистем	1.02.2020	
5.	Програмна реалізація системи	1.03.2020	
6.	Оформлення пояснювальної записки	1.05.2020	
7.	Захист програмного продукту	05.06.2020	
8.	Передзахист	08.06.2020	
9.	Захист	17.06.2020	

Студент _____ Кочкарьов Сергій Вадимович
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Метою цієї роботи є створення системи контролю доступу для мобільної точки пропуску.

Система розроблена за технічним завданням технічного директора компанії Intteks. Компанія Intteks займається розробкою и продажем охоронних систем для бізнесу. Розроблена система є відмінною від класичних охоронних систем так як вона не має статичного пункту контролю за картками-пропусками, замість цього використовується смартфон, що зчитує дані і виводить на екран дані з картки та права доступу для цієї картки с серверу, на основі цих даних контролер дозволяє або не дозволяє вхід володарю карти-пропуску. Розробка призначена для організації контролю доступу або збору відміток про переміщення людей при неможливості або недоцільності розгортання апаратних засобів обмеження доступу.

Загальний обсяг роботи: 72 сторінка, 34 ілюстрацій та (не понял) бібліографічних найменувань.

Ключові слова: охоронна система, контроль доступу, точка пропуску, картки-пропуску, додаток, система контролю, NFC мітки.

ABSTRACT

The purpose of this work is to create a access control system for smart point of control.

the system was developed according to the technical task of the technical director “Intteks” company. Intteks company develops and sells security systems for business. The developed system is different from the classic security systems because it does not have a static control point for pass cards, instead it uses a smartphone that reads data and displays data from the card and access rights for this card from the server, based on this data controller allows or does not allow entry to the holder of the pass card. The development is

designed to organize access control or collect marks on the movement of people when it is impossible or impractical to deploy hardware to restrict access.

Total volume of work: 72 pages, 34 illustrations and 14 references.

Keywords: security system, access control, checkpoint, pass cards, application, control system, NFC tags.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

NFC - Near-field communication, зв’язок на невеликих відстанях;

POS-терминал - Point Of Sale, точка продажу;

UML - Unified Modeling Language, уніфікована мова моделювання;

NDEF - NFC Data Exchange Format, формат данних для обміну за допомогою технології NFC;

СКУД - Система контролю і управління доступом;

Stack - це набір інструментів, що застосовується при роботі в проектах;

MERN – це безкоштовний MongoDB, ExpressJS, React, NodeJS стек з відкритим кодом для створення динамічних веб-сайтів та веб-додатків;

Backend – розробка стосується серверної програми програми та всього, що спілкується між базою даних та браузером;

Frontend - це практика перетворення даних у графічний інтерфейс;

JSON - JavaScript Object Notation, це відкритий стандартний формат файлу та формат обміну даними;

HTTP - The Hypertext Transfer Protocol, являє собою протокол програми для інформаційних систем з розподіленою, спільною, гіпермедіа

MVC - Model–view–controller, це схема дизайну програмного забезпечення;

Framework або фреймворк - програмне забезпечення, що полегшує розробку і об'єднання різних компонентів великого програмного проекту.

DOM - Document Object Model, міжплатформенний та незалежний від мови інтерфейс, який обробляє документ XML або HTML;

Virtual DOM - це концепція програмування, де ідеальне, або "віртуальне" представлення інтерфейсу зберігається в пам'яті та синхронізується з "реальним" DOM бібліотекою.

Dashboard - це тип графічного інтерфейсу користувача, який часто забезпечує огляди ключових показників ефективності, що стосуються певної цілі або бізнес-процесу;

IPv4 - Internet Protocol version 4. Найпопулярніший мережевий протокол

Diffing – процес порівняння двох частин будь чого;

ARIA - Accessible Rich Internet Applications;

CSS - мова стилів, яка використовується для опису подання документа, написаного мовою розмітки HTML;

REST API - Representational state transfer, представницький стан передачі - це програмний архітектурний стиль, який визначає набір обмежень, які будуть використані для створення веб-служб;

UX - User experience, досвід користування інтрефейсом;

Route – Маршрутизація означає, як кінцеві точки програми (URI) реагують на запити клієнтів

Proxy - це серверна програма або пристрій, який виступає посередником для запитів клієнтів, які шукають ресурси від серверів, які надають ці ресурси

PORT - протокол

ВСТУП

Сучасний бізнес обов'язково повинен бути оформлений найкращим чином. Безпека і автоматизація всієї діяятльності бізнесу, то це важлива умова для досягнення привабливих результатів. Тому, компанія Інттекс пропонує великий вибір привабливих варіантів, серед яких, кожен володар бізнесу з легкістю можете знайти програми та системи, котрі будуть підходити його діяльності. Користування рішеннями компанії є простим. Тому, з кожним днем, все більше підприємців думають про те, щоб скористатися сучасним обладнанням для досягнення привабливих результатів в стислі терміни.

Безпека повинна бути скрізь і в усьому. Це важлива умова для комфорту і практичності кожної сучасної людини. Тому, обов'язково потрібно враховувати всі можливі вимоги для організації безпеки свого приміщення, особливо, якщо це досить великий бізнес. Одним із елементів безпеки є турнікети, вони дають можливість контролювати всіх людей, які відвідують ваше приміщення, щоб можна було бути впевненими в тому, що все строго безпечно і надійно. Розроблені турнікети компанії Intteks використовують карти пропуску з чіпом Mifare.

Модуль NFC у сучасних смартфонах дозволяє зчитувати дані з карт-пропусків за протоколом NDEF. З розвитком комп'ютерних технологій у компанії Intteks виникла ідєя розвивати охоронні системи у тому ж напрямку, куди направлений весь світ – «смартфонізація», тобто популярізація смартфонів та іх впливу на життя. З кожним роком кількість проданих смартфонів збільшуеться на 8-10% в порівнянні з минулим роком [1]. Також

технологія NFC стає популярнішою з кожним роком. На основі цього компанія Intteks прийняла рішення розробки мобільного додатку.

При проходженні практики на підприємстві “ТОВ Intteks” була поставлена задача розробки системи мобільної точки зчитування карт-пропусків, система складається з наступних елементів: мобільний зчитувач карт-пропусків на основі технології NFC, веб-сервер з базою даних, що перевіряє наявність користувача у системі і панель адміністратора. Система призначена для охоронних систем, котрим потрібен мобільний зчитувач карт-пропусків.

Основна ціль створення такого додатку: простий та швидкий спосіб зчитування данних з карт-пропусків. Додаток дозволяє створювати мобільні точки пропуску не прив’язуючисть до місця – тобто у ситуаціях, коли потрібно створити тимчасову точку пропуску не використовуючи статичних турнікетів пропуску, також прискорює відладку та перевірку працездатності карт-пропусків і корректної роботи турнікетів.

Мобільний додаток має можливості: зчитування даних з захищеної області за протоколом NDEF з карт торгової марки Mifare, відправка даних на веб-сервер та отримання результату з інформацією про володаря відсканованої карти-пропуску. Веб-сервер у свою чергу має повну історію взаємодії з системою, інтерактивну карту та панель редагування прав доступу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Причини розробки системи

Сфери використання мобільного зчитувача:

1. Організація контролю доступу або збір відміток про переміщення людей при неможливості або недоцільності розгортання апаратних засобів обмеження доступу, наприклад, тимчасових точок доступу, аварійних точок доступу, віддалених територіально точок доступу.

2. Контроль права переміщення і збір відміток про переміщення людей через контрольовані транспортні проїзди в автобусах і інших автомобілях: на автобусі або автомобілі переміщається неколько людей, класично їх би довелося вивести з автобуса / автомобілі і змусити пройти через транспортну прохідну, на великих підприємствах, де великі території, людей часто завозять на територію в автобусі і везуть прямо до цеху.

3. Ідентифікація людей на контрольованій території рухливими групами охорони.

4. Транспорт, прикладом в Києві є верифікатори в наземному транспорті для оплати проїзду.

Додаткові причини розробки:

- Тестування потужності напрямку мобільних точок контролю для бізнесу.
- Використання та аналіз можливостей технології NFC.
- Побудова систем, додатків, сервісів з використанням NFC на основі досвіду реалізації цього проекту.

1.2 Аналіз сучасних тенденцій

Глобальна популяризація смартфонів у світі дає значну можливість для зростання мобільних платежів з підтримкою NFC. Прикладом систем для оплати товарів смартфоном є: Google pay – платіжна програма на базі смартфонів Android, має можливість відправляти грощі друзям та рідним.

Цей додаток також дозволяє користувачам використовувати більшість дебетових та кредитних карт великих брендів, таких як картки Discover, American Express та багато іншого. Більш того, ви також можете використовувати картки, видані великими банками, такими як Citi, Bank of America та Wells Fargo. Також одна з найпопулярніших систем оплати - Apple Pay. Apple Pay була представлена у 2014 році. Вона доступна для iPhone 6 та новіших версій, а також iPad (iPad mini 3 та iPad Air 2 та новіших версій) та Apple Watch, що дозволяє клієнтам здійснювати "безконтактні" платежі в магазинах. Функція Touch ID підвищує безпеку цієї програми. Apple Pay також постачається з токенізацією, а це означає, що iPhone передає маркету торговець замість реквізитів, таких як номер картки. Тож, навіть якщо зловмисникові вдасться викрасти маркер із платіжної системи продавця, він не зможе отримати інформацію про карту, на відміну від систем оплати на операційній системі Android. Перелік інших менш популярних систем оплати: LifeLock Wallet, MasterCard PayPass, Visa payWave, Square Wallet.[1]

1.2.2 Переваги технології NFC як засіб оплати платежів:

- Швидкість – у порівнянні з платежами, здійсненими готівкою, чіпом чи PIN-картами, платіжні системи NFC набагато швидше. Це тому, що користувачеві потрібно просто торкнутися картки NFC або розмістити смартфон з NFC біля зчитувача. Для продавця це також зручно, оскільки їм не потрібно чекати, коли клієнти введуть свій PIN-код або шукають готівку. За допомогою NFC виплати можна здійснити протягом декількох секунд. Довгі черги в магазині негативно впливають на покупців, оскільки вони можуть піти, нічого не купуючи. Якщо покупець щоразу, коли він відвідує магазини, знаходить довгі черги, він може позначити магазин як переповнений і поїхати кудись ще. Технологія NFC може підвищити швидкість оформлення каси для

всіх клієнтів. Це робить черги коротшими та покращує враження як для власників магазину, так і для покупців.

- Зручність - Однією з найбільших переваг платежів через NFC є зручність. Технологія NFC спростила користувачам здійснення платежів за допомогою своїх планшетів та смартфонів. Загальний процес оплати також простий у використанні та розумінні. За допомогою цього користувач може здійснювати транзакції простим дотиком або кількома натисканнями на смартфоні.
- Універсальність - NFC дуже універсальний за своєю природою, оскільки охоплює великий спектр різних галузей та послуг. NFC дозволяє користувачеві здійснювати платежі в різних місцях, з яких небагато з них:
  - Паркомати, вже використовуються в Україні для оплати паркувальних місць
  - Квитки в кіно
  - Платіжні термінали, вже впровадженим прикладом є термінал Ibox, де замість ручного вводу номеру картки для її поповнення можна її прикласти до зчитувача.
  - Торгівельні платежі у магазинах
  - Нагороди, прикладом є бонусні мобільні додатки ресторанів.
  - Банківські картки
  - Оплата квитків, вже впровадежним прикладом є турнікет входу у київському метро, що використовують Apple і Google pay.
  - Енергоспоживання, окрім NFC, існує багато технологій, які дозволяють пристроям спілкуватися із системою точки обслуговування (POS термінали). Технологія Bluetooth - приклад, що забезпечує подібну перевагу. Однак головна перевага NFC полягає в тому, що йому потрібно менше енергії для завершення транзакції. Це робить NFC кращим вибором для торговців, які

турбуються про те, що їх носіння та смартфон будуть виснажені через численні транзакції.

1.2.3 Недоліки використання NFC:

Висока ціна технонології. Технологія NFC має безліч переваг; однак ці переваги пов'язані з витратами. І ця вартість занадто висока, щоб деякі компанії могли собі дозволити. Такі компанії, як Starbucks, успішно інтегрували технологію NFC у свою систему; однак, для менших компаній інтеграція NFC може спричинити труднощі у підтримці їх обороту та збільшення прибутку. Це тому, що вартість встановлення програмного забезпечення та обладнання разом із наймом техніків дуже висока.

1.1.4. Актуальність технології у сучасному світі.

Протягом багатьох років ми спостерігали величезний технологічний прогрес у галузі фінансів та платежів. Втручання технології повністю змінило спосіб здійснення платежів сьогодні. Що стосується Інтернет-банкінгу, мобільного банкінгу та платежів за мобільний гаманець, ми більше не залежимо від грошових коштів.

З часом ми спостерігаємо нові технології, які використовуються для покращення досвіду платежів як ніколи. І однією з таких технологій є NFC.

Платежі за NFC, які також відомі як безконтактні платежі, придбали величезну популярність у всьому світі. У 2015 році розмір глобального ринку NFC оцінювався у $ 4,80 млрд., Ринок збільшиться в 10 разів до 2024 року, орієнтовна оцінка - 50 мільярдів доларів, це зображено на рисунку рисунку 1.[2]

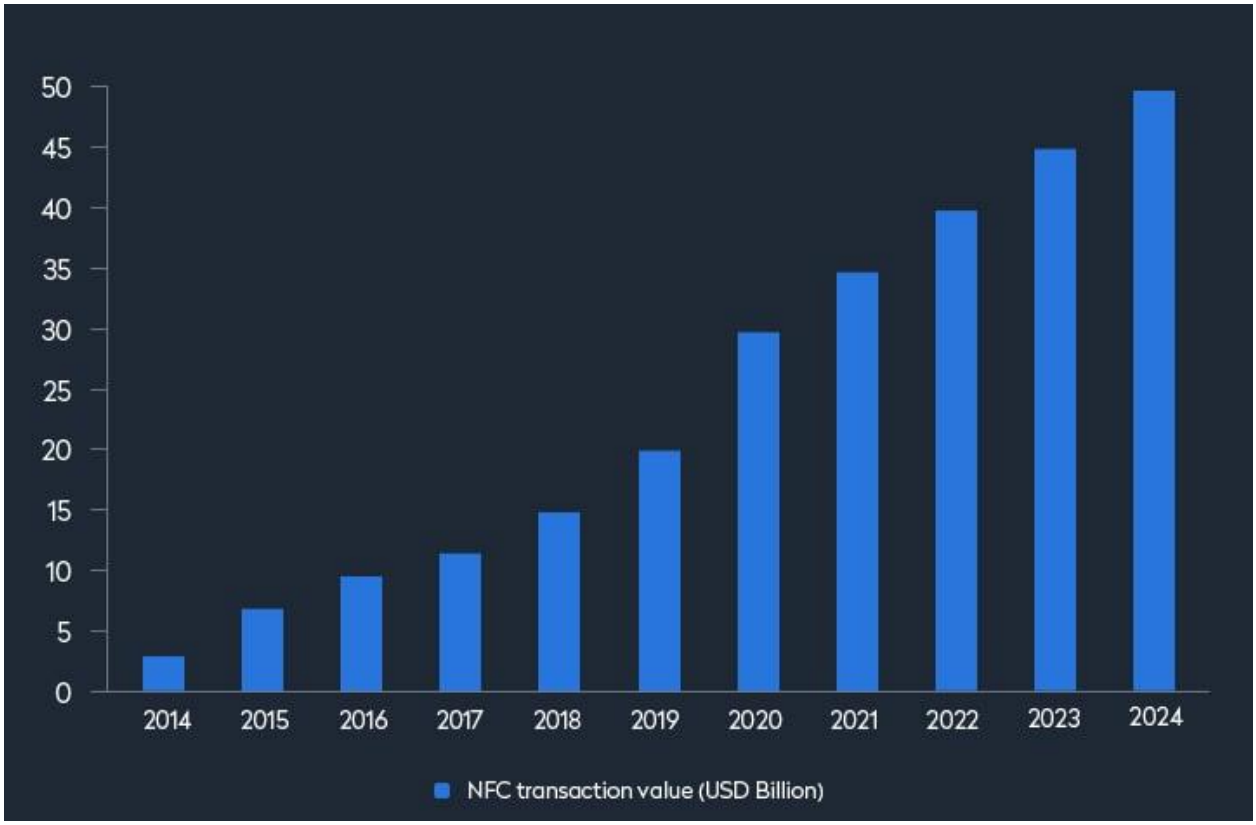


Рисунок 1 Популяризація технології NFC у світі

1.2.5. Висновки щодо технології NFC:

Завдяки величезним перевагам та розширеним можливостям, технологія NFC стане популярною в найближчі роки. Ви не повинні дивуватися, якщо новим нормою стає прийняття чи здійснення платежів безконтактними методами. NFC - це швидкий, зручний та безпечний спосіб здійснення платежів, які можуть бути прийняті як користувачами, так і дрібними та середніми роздрібними торговцями. Це правда, що недоліків технології NFC мало, особливо коли мова йде про безпеку; проте з часом ця технологія вдосконалиться, зробивши її більш безпечною для платежів.

1.3 Опис предметної області

Предметною областю моєї розробки є: охоронні системи - СКУД і сучасні технології що використовуються для забезпечення охорони або вдосконалення роботи охоронних систем, а саме - NFC.

У школах, офісах, магазинах, на парковках, в житлових будинках, щоб забезпечити безпеку закрити доступ небажаним особам, застосовується система контролю і управління доступом. СКУД призначена не тільки забезпечити, спільно зі службою охорони, неможливість проникнення сторонніх на об'єкт, що охороняється, його безпеку, а й істотно полегшити роботу відділів кадрів підприємств, бухгалтерії. Так як вона автоматизує облік робочого часу. Сучасна СКУД є набором охоронних технічних засобів для відеоспостереження і запобігання несанкціонованого проникнення, керованих дистанційно, об'єднаних в єдину систему. Для кожної такої системи, її пристроїв, спеціально розробляється програмне забезпечення.

1.3.1 Основні елементи і технічні пристрої СКУД

Залежно від сфери застосування в систему можуть входити різні компоненти обладнання, розглянемо стандартний набір технічних і програмних засобів:

- контролери;
- ідентифікатори;
- зчитувачі;
- виконавчі механізми;
- відеоспостереження;
- автоматизовані робочі місця, програмне забезпечення;

Основним елементом, електронним мозком системи є контролер. Саме він, проаналізувавши підсумки від зчитувача відомості, приймає рішення про надання доступу або відмову в ньому. Дає команду виконавчим механізмам відкрити двері, турнікет, шлагбаум або заблокувати їх. Відомості про режим роботи, конфігурації системи, список співробітників з категоріями доступу - все це необхідно для роботи цього розумного пристрою. Інформація також зберігається в ньому. Працюючі автономно контролери застосовуються там, де тільки є одна точка доступу до об'єкта. Якщо таких точок багато, застосовуються мережеві контролери.

Для прийняття рішення про можливість допуску конкретної людини або автомобіля на об'єкт, що охороняється, потрібно його ідентифікувати в системі. Для цього використовують різні види ідентифікаторів, зчитувачів до них. Ідентифікатором може бути брелок, безконтактна картка, браслет, мобільний телефон. Один пристрій дасть користувачеві доступ на парковку, в офіс або квартиру, можливості використання міських сервісів. У пристрій можуть бути додатково вбудовані транспортні карти, бонусні карти для програм лояльності. Контролер, проаналізувавши отримані від зчитувача відомості, надати їм доступ або відмовляє в ньому. Дає команду виконавчому пристрою відкрити або заблокувати прохід.

Щоб вчасно відкривати прохід або навпаки, надійно його блокувати, застосовуються керовані контролером виконавчі пристрої. Це різні електронно-механічні, електронні замки, шлагбауми, турнікети, магнітні засувки, автоматичні ворота. Замки і замикаючі пристрої відкриваються або блокуються подачею короткочасного відмикає / блокуючого імпульсу. До виконавчих пристроїв також відносяться система віддаленого відкриття дверей, відеодомофон або інше обладнання. Потрібно обов'язково використовувати джерело резервного живлення, щоб раптові збої електропостачання не вплинули на надійність роботи системи.

Для оптимізації роботи СКУД рекомендується поєднувати її з системою відеоспостереження. На екрані охоронця можна бачити входи і виходи, двері, ворота, турнікети. Оперативно зреагувати на потенційну небезпеку, заблокувати будь-який вхід або вихід можна прямо з робочого місця чергового. Все позаштатні ситуації будуть зафіксовані на відео.

Ці пристрої не можуть працювати без спеціального програмного забезпечення. Для кожного пристрою компанії-виробники розробляють своє програмне забезпечення. Використання сторонніх програмних продуктів буде порушенням авторських прав. З СКУД в комплекті йде вже встановлене на всі пристрої ПО.

1.3.2 Основними функціями, що визначає призначення СКУД, є:

- перевірка допуску та контроль доступу. Здійснення суворого контролю доступу людей і техніки на об'єкт, що охороняється. Обмеження проходу в приміщення або проїзду на певні території в суворій відповідності з рівнем допуску;
- контроль переміщення співробітників і облік робочого часу. Вхід і вихід кожного співробітника і відвідувача фіксує система, в базу даних вносяться необхідні зміни, зберігаються в архіві. Для бухгалтерії та відділів кадрів це суттєво полегшує контроль трудової дисципліни співробітників, дотримання режиму роботи підприємства. Перевірити, скільки годин відпрацьовано у вихідні та святкові дні, у нічний час, завдяки цій системі нескладно. Спроби співробітників проникнути в приміщення, що не відповідають їх рівню доступу, також відзначає система;
- охорона об'єкта. Заборона проходу і проїзду на об'єкт, що охороняється через точки доступу всіх не володіють відповідним допуском людей, а також технічних засобів. Контроль допуску зареєстрованих в системі постійних користувачів і тимчасових відвідувачів об'єкта охорони. Реєстрація всіх входів і виходів, в'їздів і виїздів, збереження даних в архіві. Дистанційне керування точками доступу онлайн, можливість перекрити і заблокувати всі входи і виходи в одну мить;
- спільне застосування СКУД і пожежної сигналізації, оперативний виклик охоронних або пожежних служб в разі спроби несанкціонованого проникнення на територію, що охороняється, злому воріт і дверей, а також при загорянні пожежі. При пожежі важливо також відкрити доступ до всіх приміщень для успішної ліквідації вогнища загоряння.

Для різних типів об'єктів застосовують різні види СКУД. Маленький офіс або житловий будинок, велике підприємство, торговий центр або школа - для кожного з цих об'єктів потрібна своя, особлива система, хоч і схожа на інші.

Основні види контрольованих об'єктів:

- великі корпорації або підприємства;
- невеликі офіси, магазини, готелі;
- парковки;
- школи;

1.4 Постановка задачі

Розроблювана складається с двох частин: мобільного додатку та веб-додатку. Веб-додаток у свою чергу складається з двох частин: frontend і backend додатки, що працють одночнасно і мають між собою клієнт-серверну архітектуру.

Етапи технічної розробки системи

1.4.1 Розробка мобільного додатку:

- Створення загальної концепції додатку;
- Розробка базового додатку за допомогою мови React Native, а саме: розробленка інтерфейсу програми, розробка основного модулю зчитування даних за технологією NFC;

1.4.2 Розробка серверної частини:

- Розробка кліент-серверну архітектуру системи;
- Розробити серверну частину на основі серверної платформи Node.js з використанням фреймфорку Express.js;
- Встановити хмарну систему БД MongoDB для додатку.

1.4.3 Розробки панелі адміністратора:

- Розробка frontend серверу за допомогою React фреймворку;

1.4.3 Заповнення бази даних за наступними полями:

Вхідними даними до системи дані з NFC міток, що використовуються як перепустки. Дані реєструються у системі під час сканування мітки, разом з даними мітки до бази даних також записується додаткова інформація, а саме запис події зчитування, геолокація події зчитування.

Тобто для користувача мобільного додатку вхідною інформацією є тільки картка-зчитувач. Для адміністратора вхідна інформація складається з трьох основних таблиць бази даних:

- Геолокації – містить інформацію про розташування усіх сканувань міток
- Мітки – містять всю інформацію про мітку і особу, що її використовує, додатково у цій таблиці встановлюються права доступу.
- Журнал подій – зберігає дані про усі події у системі, це можуть бути: зчитування міток, рєєстрація нової мітки у системі, зміна прав певної мітки, завантаження фото користувача для картки-пропуску.

Система повинна обробляти наступні види інформаційних даних, що будуть подані у виді об’єктів, а саме:

Вхідна інформація:

- Мітка:
  - Id
  - Ідентифікатор мітки
  - Ім’я
  - Фото
  - Права доступу
- Журнал подій:
  - Id
  - Ідентифікатор мітки
  - Тип події
  - Дата
  - Автор
  - IP-адреса
- Геолокації:

- Id
- Ідентифікатор мітки
- Дата
- Координати, містять у собі довготу і широту

1.4.4 Обгрунтування і визначення проблем, що вирішує розробка – подано у таблиці 1.1.

Таблиця 1.1

Визначення проблеми

Проблема	Відсутність ПЗ що має можливість сканувати дані з карт-пропусків без статичного турнікету.
Кого чіпляє	Підприємців, установників охоронних систем.
Наслідки проблеми	Відсутність можливості робити недорогі переносні точки пропуску.
Успішне вирішення	Можливість встановлювати переносні пункти пропуску без встановлення дорогих і статичних систем пропуску. Можливість швидкого тестування статичних турнікетів до вводу їх в єксплуатацію.

1.4.5. Позіціонування. Визначення позиції додатку та мети створення описано в таблиці 1.2.

Таблиця 1.2

Позиціонування продукту

Необхідно	Систему контролю доступу для мобільної точки пропуску.
Назва продукту	Назва мобільного додатку - NFCreader, зчитувач даних с NFC міток.
Деталі	Використовуване програмне забепечення:

	графічний інтерфейс та основна кліентська частина мобільного додатку розроблена за допомогою: React Native з використанням бібліотеки react-native-nfc-manager. Серверна частина розроблена на основі фреймворку мови Javascript – Node.js. Основний фреймворк використовуваний у Node.js – Express.js. Для бази даних була використана не реляційна база даних – MongoDB, робота з якою на серверній частині проходить з фреймворком Mongoose.
Огляд аналогів	Було оглянуто велику кількість веб-сайтів по обранню та моделюванню теплиць, але на жодному з них не спостерігався функціонал який буде реалізо-вано в нашій системі.
Розроблюваний продукт	Дає можливість зчитувати дані з карт-пропусків за допомогою смартфону з NFC чипом. Продукт розроблений на універсальному фреймворку, що дозволяє встановити додаток на різні операційні системи: IOS, Android. Серверна частина розроблена за допомогою React. Візуальна частина для панелі адміністратора серверної частини розроблена за допомогою Material UI. Продукт має простий і зрозумілий інтерфейс.

1.4.6 Опис користувачів. Система має 2 типи користувачів: адміністратор веб-серверу, користувач додадку. Детальний опис і відповідальність кожного зображено у таблиці 1.3.

Таблиця 1.3

Таблиця опису користувачів системи

Преставник	Адміністратор	Користувач додатку
Опис	Має доступ до бази даних, може змінювати права доступу для первних пропусків, додавати, видаляти дані пропусків, може переглядати історію подій.	Може зчитувати карти-пропуски та отримувати інформацію від серверу. пропуску.
Відповідальність	Додає, видаляє, змінює права доступу для пропусків, історії подій.	Має можливість відмічати подію пропуску і відправляти на сервер для карти-

1.5 Огляд існуючих рішень

1.5.1 Аналіз потреб. Аналізуючи потреби компанії Intteks було визначено проблеми:

- відсутність альтернативних рішень створення мобільної точки пропуску;
  складність встановлення статичних систем пропуску;
- швидкість розгортання системи пропуску;

Для вирішення зазначених проблем було створено значну кількість застосунків у мобільному додатку системи.

1.5.2 Огляд альтернативних систем. Альтернативними системами є статичні точки контролю. Основним прикладом альтернативи буде рішення компанії Intteks для бізнес центрів, система доступна за адресою http://intteks.com.ua/typical-solutions/biznes-tsentry.

Будівля оснащується багаторівневої інтегрованої системою безпеки, розробленої на базі обладнання «КОДОС». Безпосередньо будівлю офісного центру обладнується системою контролю доступу, охоронно-пожежною сигналізацією і системою відеоспостереження. На вході в будівлю організовується бюро перепусток для відвідувачів, а прилегла автопарковка оснащується системою контролю проїзду.

Завдання, які вирішуються:

- Контроль несанкціонованого проникнення в приміщення офісного центру;
- видача індивідуальних пропусків для орендарів;
- організація бюро перепусток для відвідувачів і гостей з можливістю внесення їх фотографій і особистих даних в базу;
- розмежування доступу співробітників і відвідувачів в різні офісні приміщення, контроль за їх переміщеннями;
- при необхідності - облік робочого часу персоналу офісного центру.
- Індивідуальна або централізована постановка / зняття з охорони приміщень орендарями.
- Охоронно-пожежна сигналізація в офісах.
- Автоматизація робочих місць співробітників служби безпеки.

Переваги:

- Підвищення загального рівня безпеки офісного центру за рахунок припинення несанкціонованого доступу в приміщення.
- Повний контроль за поточною ситуацією в офісному центрі за рахунок автоматизації видачі пропусків, обліку відвідувачів і доступу автотранспорту на парковку.
- Можливість оперативного реагування на позаштатні ситуації, відстеження яких ведеться в режимі реального часу.

Недоліки:

- потребує велику свободної площі;

– немає можливості самойстійно змістити або частково переформувати систему;
– дуже висока ціна;
– складається з великої кількості окремих частин, що потребують підтримки у ситуації виходу з робочого стану;
– відсутність можливості переміщати систему;
– складність впровадження і використання для малих та середніх організацій;

Загальний вигляд системи зображений на рисунку 1.

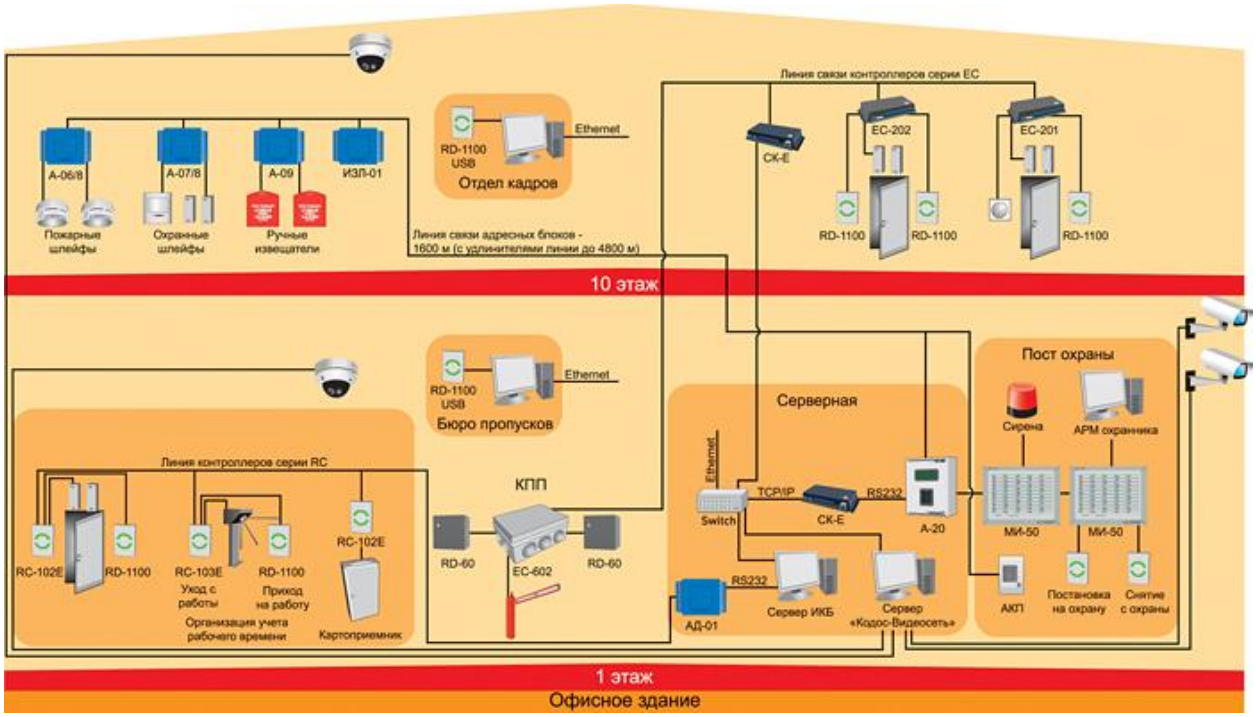


Рисунок 1 - Загальний вигляд системи [5]

На рисунку 1 зображено:

– пост охрани;
– серверна кімната;
– відділ кадрів;

2 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

Моделювання предметної області - це моделювання об'єктів обліку шляхом їх класифікації. Класифікація завжди суб'єктивна і тому вимагає вказівки на суб'єкт, який провів класифікацію.

Для модулювання предметної області було обрано мову UML.

UML - це загальноприйнята, моделююча мова в галузі інженерії програмного забезпечення, яка покликана забезпечити стандартний спосіб візуалізації дизайну системи. Ця мова активно розвивається у наш час.

Метою створення уніфікованої мови моделювання було: прагнення стандартизувати розрізнені системи та підходи до розробки програмного забезпечення, прагнення оптимізувати процес розробки програмних систем. Він був розроблений Грейді Бучем, Іваром Джейкобсоном та Джеймсом Румбо в Rational Software у 1994–1995 рр.

UML відображає наступні елементи:

- будь-яку діяльність робочого місця;
- окремі компоненти системи;
- як компоненти можуть взаємодіяти з іншими компонентами програмного забезпечення;
- як працюватиме система в цілому;
- як взаємодіють об'єкти з іншими компонентами та інтерфейсами;
- зовнішній інтерфейс користувача;

2.1 Різниця між UML та діаграмами системи.

Діаграма - це часткове графічне зображення моделі системи. Набір діаграм не повинен повністю охоплювати модель, а видалення діаграми не змінює модель. Модель також може містити документацію, яка керує елементами моделі та діаграмами (наприклад, випадки письмового використання).

Набір діаграм не повинен повністю охоплювати модель, а видалення діаграми не змінює модель. Модель також може містити документацію, яка

керує елементами моделі та діаграмами (наприклад, випадки письмового використання).

Діаграми UML представляють два різні погляди на системну модель:

- Статичний або структурний погляд: підкреслює статичну структуру системи з використанням об'єктів, атрибутів, операцій та зв’язків. Вона включає в себе діаграми класів і складові діаграми структури.
- Динамічний або поведінковий погляд: підкреслює динамічну поведінку системи, показуючи співпрацю між об'єктами та зміни внутрішніх станів об’єктів. Цей вид включає діаграми послідовності, діаграми активності та діаграми стану машини.

Моделі UML можна обміняти серед інструментів UML за допомогою формату обміну метаданими XML (XMI).

В UML одним із ключових інструментів моделювання поведінки є модель використання випадків, викликана OOSE. Випадки використання - це спосіб вказати необхідні звичаї системи. Зазвичай вони використовуються для охоплення вимог системи, тобто того, що система повинна робити.

2.2. Структурні діаграми. Вони показують статичну структуру системи та її частин на різних рівнях абстракції та реалізації та те, як вони пов'язані між собою.

Елементи на структурній діаграмі представляють змістовні поняття системи, і вони можуть включати абстрактні, реальний світ та концепції реалізації, є сім типів структурної діаграми наступним чином:

1. Діаграма класів

Діаграма класів - це центральна технологія моделювання, яка проходить майже через усі об'єктно-орієнтовані методи. Ця діаграма описує типи об’єктів у системі та різні види статичних зв’язків, що існують між ними.

Відносини:

- Асоціація - представляє відносини між екземплярами типів (людина працює в компанії, компанія має ряд офісів.

– Спадкування - найочевидніше доповнення до діаграм ER для використання в ОО. Він має безпосередню відповідність спадку в ОО-дизайні.
– Агрегація - агрегація, форма об’єктної композиції в об’єктно-орієнтованому дизайні.

На рисунку 1 зображено діаграму класів для нашого backend додатку.

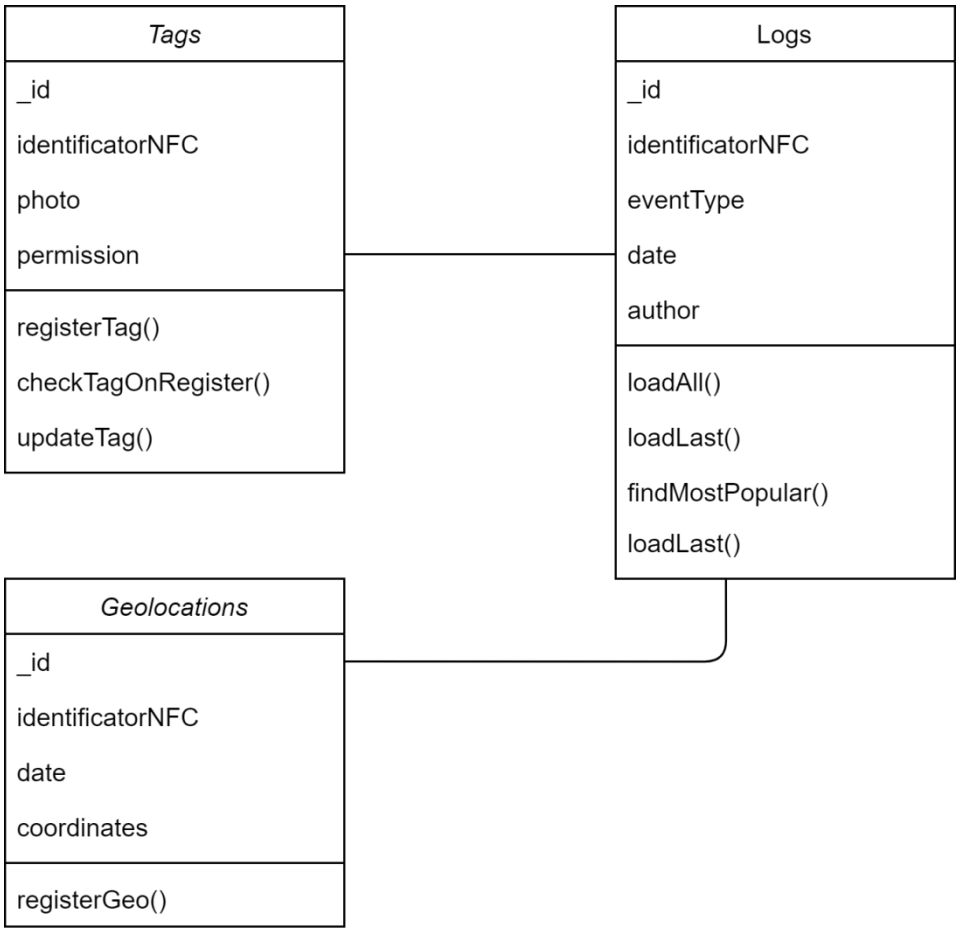


Рисунок 1. Діаграма класів

На рисуноку 1 зображена діаграма класів для програмного продукту. Реаліцазія цього

2. Діаграма компонентів. У UML діаграма компонентів зображує, як компоненти з'єднані між собою для формування великих компонентів або програмних систем. Він ілюструє архітектуру програмних компонентів та залежності між ними. Ці компоненти програмного забезпечення, включаючи компоненти часу виконання, виконувані компоненти, також компоненти вихідного коду.

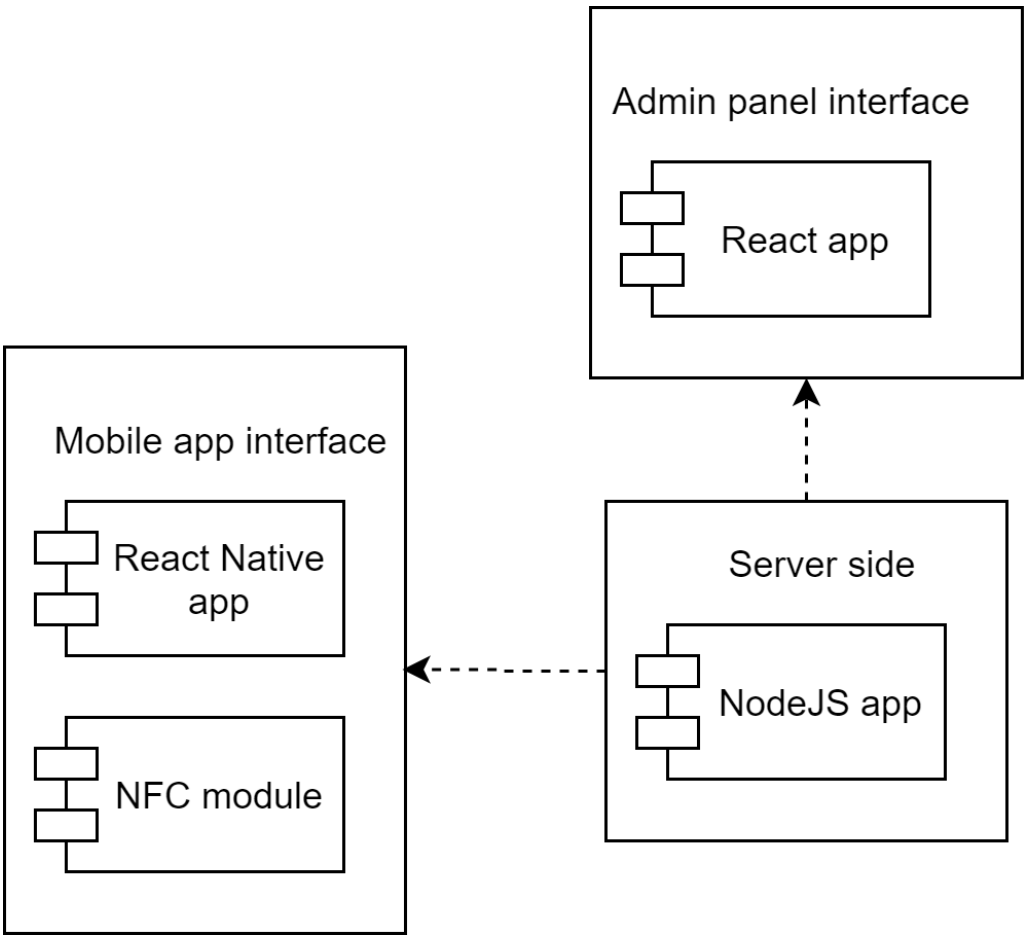


Рисунок 2. Діаграма компонентів

На діаграмі компонентів з рисуноку 2 зображено складові частини системи: веб-сервер що використовує мову Node.js, мобільний додаток на мові React Native та панель адміністратора.

Кожен с серверів використовує наступні порти для своєї роботи:

- Порт 3000 -веб-сервер;
- Порт 4000 – панель адміністратора;
- Порт 8081 – відладка та встановлення мобільного додатку на пристрій з якого проходить тестування;

3. Діаграма розгортання допомагає моделювати фізичний аспект об'єктно-орієнтованої програмної системи. Це структурна схема, яка показує архітектуру системи як розгортання (розповсюдження) програмних артефактів для цілей розгортання. Артефакти являють собою конкретні

елементи у фізичному світі, які є результатом процесу розвитку. Він моделює конфігурацію часу виконання у статичному вигляді та візуалізує розподіл артефактів у програмі. У більшості випадків це включає моделювання апаратних конфігурацій разом із компонентами програмного забезпечення, що працювали на них.

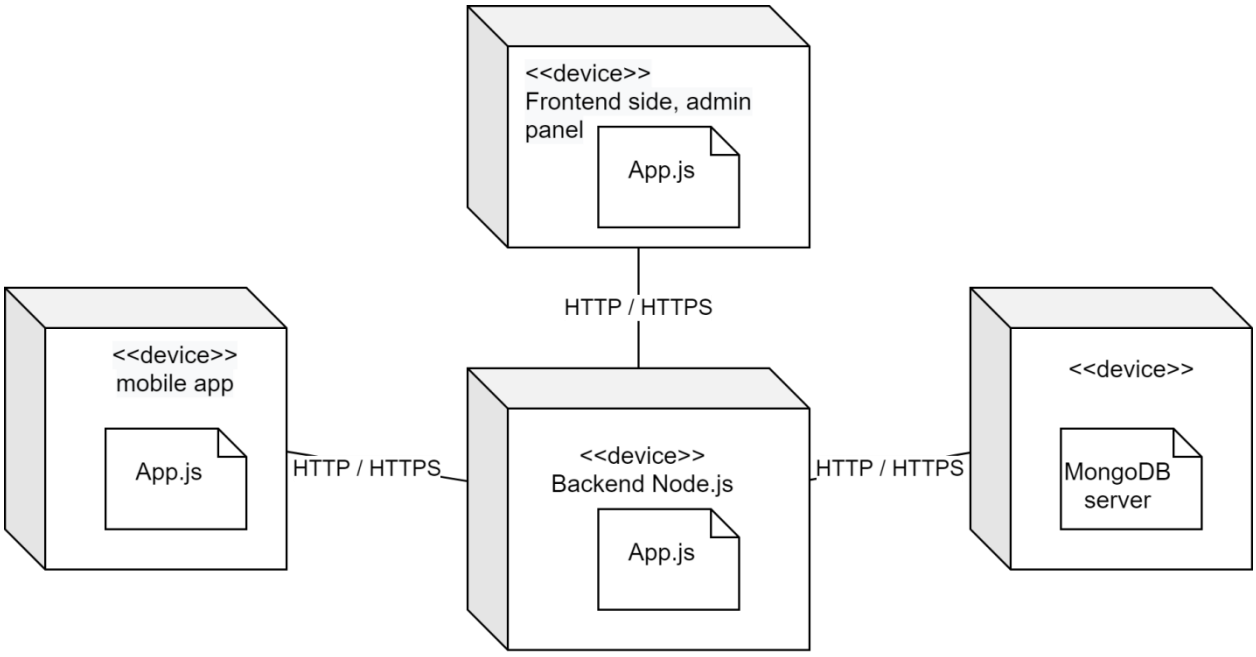


Рисунок 3. Діаграма розгортання [6]

4. Діаграма об’єктів. Показує, як екземпляри об'єктів у вашій системі взаємодіють між собою у певному стані. Тобто, діаграма об'єктів UML може розглядатися як подання про використання класів у певному стані.

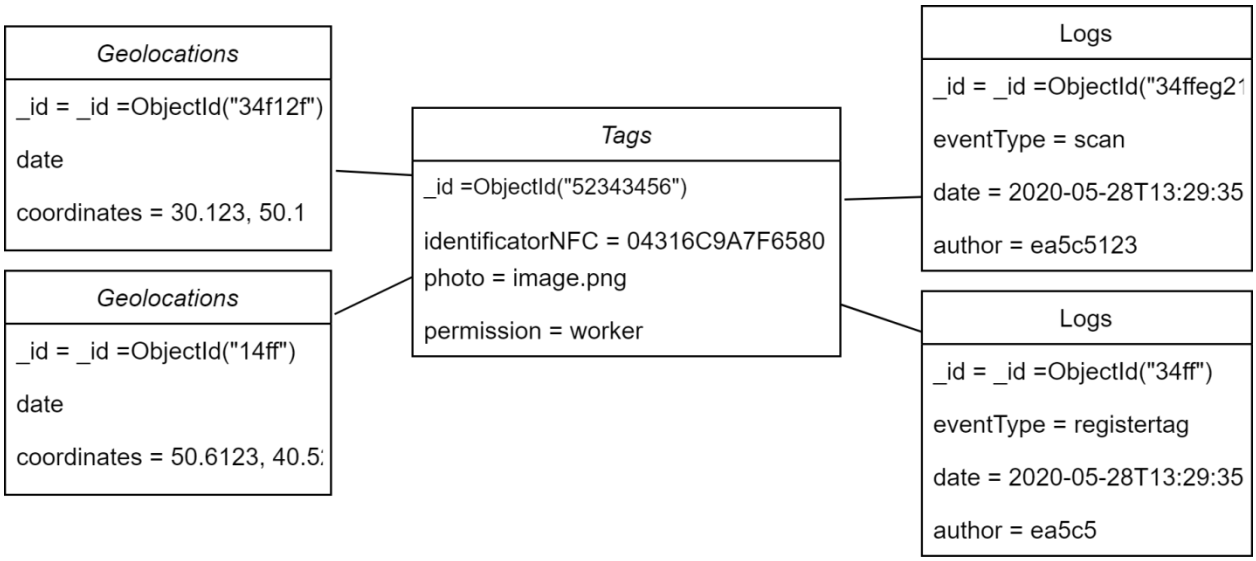


Рисунок 4. Діаграма об’єктів

На рисунку 4 зображена діаграма об’єктів, вона зображує, що 1 мітка може мати декілька записів у таблицях Геолокацій та у Журналу подій.
5. Use Case або діаграма прецедентів. Модель використання випадку описує функціональні вимоги системи з точки зору використання. Ця модель призначення для відображення усіх випадків користування з оточенням системами. Оточенням системи є суб'єкти. Випадки використання дозволяють співвіднести те, що вам потрібно з системою, і спосіб її забезпечення для цих потреб.

Прикладом є меню ресторану: дивлячись на меню, ви знаєте, що вам доступне, окремі страви, а також їх ціни. Ви також знаєте, яку кухню подає ресторан: італійська, мексиканська, китайська тощо. Переглядаючи меню, ви отримуєте загальне враження від обіднього досвіду, який чекає вас у цьому ресторані. Фактично меню "моделює" поведінку ресторану.

Рисунок 5. Діаграма Use Case

На рисунку 5 зображено приклад Use Case діаграми. Контролер взаємодіє тільки с мобільним додатком, а адміністратор додатково має доступ до веб-панелі адміністратора.

2.2 ВИСНОВКИ ДО РОЗДІЛУ 2

Програмне забезпечення, яке розробляється для представників малого та середнього бізнесу використовує великі апаратні ресурси. Створення системи повинно бути задокументовано та правильно спроектовано. Було досліджено основну концепцію UML, оцінено потужність інстументу та розроблено основні діаграми.

У результаті проектування було створено основні діаграми, на яких будується програмне забезпечення: діаграма класів, діаграма прецедентів, діаграма об’єктів, діаграма компонентів, діаграма розгортання.

3. ЗАСОБИ РОЗРОБКИ СИСТЕМИ

Для розробки було обрано стек технологій MERN, що означає Mongose, Express, React, Node JS. Стек MERN - це комбінація перерахованих вище технологій, основана на JavaScript, що використовується для створення сучасних веб-додатків. Цей стек технологій забезпечує повну розробку backend та frontend частин. Javascript мова програмування, яка використовується повсюдно, як для коду на стороні клієнта, так і для коду на стороні сервера. Завдяки одній мові, на кожному рівні розробки, розробнику не треба вивчати іншу мову програмування для редагування різних частин проекту. На відміну від інших стеків технологій, у цій розробник не повинен вивчати інші мови програмування, завдяки JavaScript, розробникам потрібно лише володіти JavaScript та JSON. Загалом, використання стека MERN дозволяє розробникам створювати високоефективні веб-програми. Повний стек технологій зображений на рисунку 1.

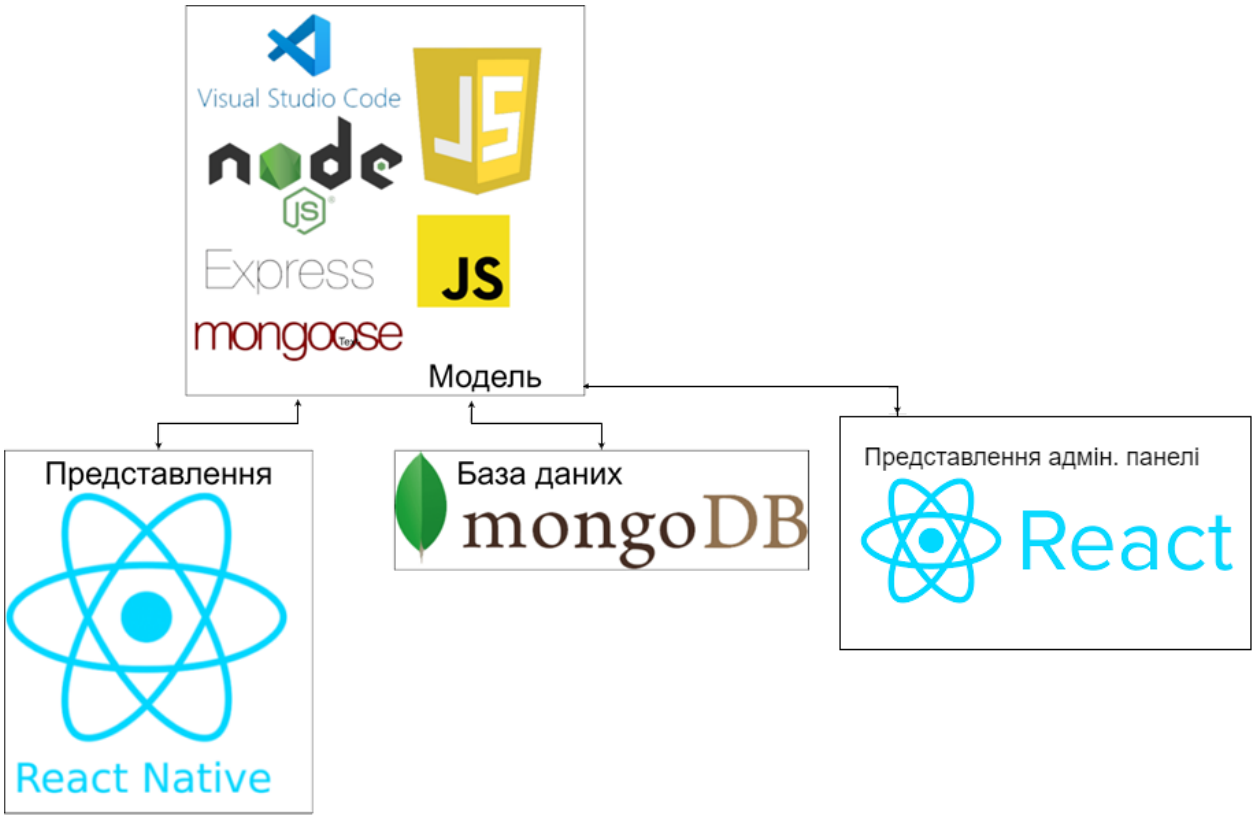


Рисунок 1. Повний стек технологій

Відповідно зображенню на Рисунку 1.1 для реалізації додатку були використані наступні засоби реалізації:

- Середовище Vscode (Visual Studio Code) – потужне середовище розробки
- Node.js – програмна платформа для розробки веб-серверу
- Express.js – фреймворк для розробки веб-додатків для Node.js
- Javascript – основна мова розробки програмного продукту
- Mongoose JS – основний модуль для роботи з MongoDB
- MongoDB – документоорієнтована система управління базами даних, що не потребує опису схем таблиць
- React Native – фреймворк для розробки мобільних додатків розроблений компанією Facebook;
- React – фреймворк, що призначений для побудови веб-інтерфейсу для користувача;
- Git – для контролю версій програмного продукту

Вибрані технології мають кросс-платформену підтримку. Середовищем розробки було обрано VScode через його швидкодію. Мовою програмування був обраний Javascript через слабку динамічну типізацію, має прототипний стиль типізації і також розробка серверної частини для Node.js ведеться на цій же мові.

MongoDB була обрана як основна база даних через модель даних документів, що дозволяє вести швидку розробку продукту, горизонтальна архітектура масштабування дозволяє підтримувати великі обсяги даних.

3.1 Середовище розробки Vscode

Visual Studio Code – «легкий» та кросс-платформений редактор коду, призначений для веб-розробки, має вбудований відладчик та інструменти для контролю версій Git.

У редакторі коду встановлені наступлені розширення, зображено на рисунку 1:

- ReactJS code snippets – для швидкої побудови компонентів React коду
- Live Sass Compiler – для автоматичного перекомпонування Sass або Scss файлів у CSS
- ES7 React/Redux/GraphQL/React-Native snippets – для швидкої побудови React-native компонентів
- React Native tools – для швидкої відладки та запуску React Native компонентів

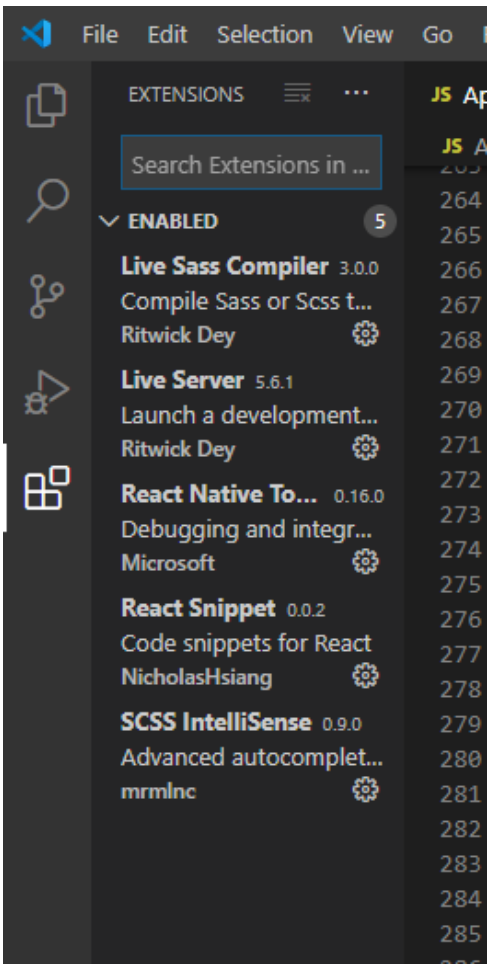


Рисунок 1

3.2. Програмна плафторма Node.js

Node.js побудована на V8 двигуні, що транслює Javascript у машинний код. Платформа перетворює вузькоспеціалізовану мову JS у мову широкого напряму. Також дозволяє взаємодіяти з Input/Output через власний API, є можливість додавати у проект сторонні бібліотеки написані на інших мовах за допомогою Node package manager.

Виконує роль веб-серверу. Node є дуже потужним і швидким інструментом. Потужність цієї платформи здобута за допомогою взаємодії V8 Javascript двигуна та

Libuv бібліотеки, що забезпечує підтримку асинхронного вводу-виводу на основі Event Loop (цикл подій) , написана на C++. У Node.js немає прямої взаємодії з C++ і libuv.

Демострацією внутрішньої роботи платформи є Рисунок 2

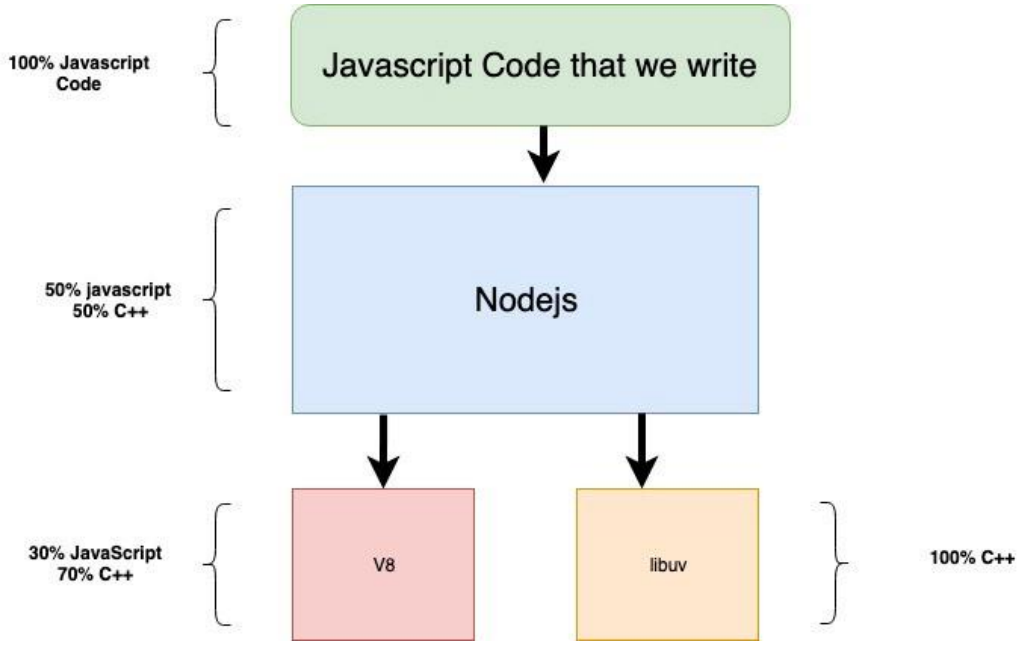


Рисунок 2 – Робота Node.js всередині

Бібліотеки як http, fs, crypto дуже узгоджені з API платформи і зсилаються на внутрішні функції libuv, через це користувач не має доступу до внутрішнього коду мови C++, приклад взаємодії зображений на Рисунку 3

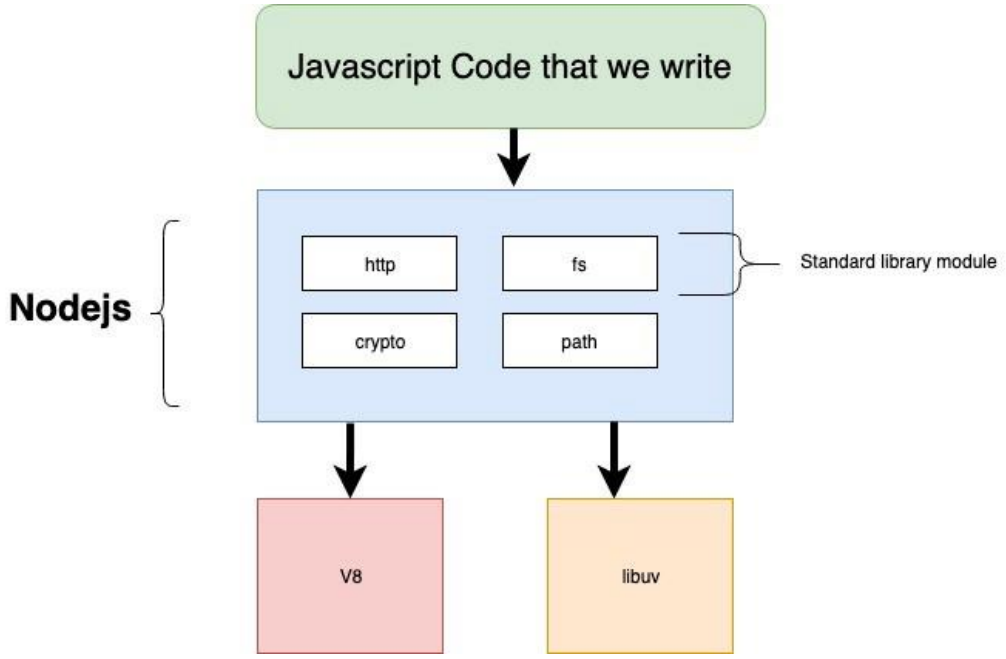


Рисунок 3 – взаємодія бібліотек з libuv

Event loop дозволяє виконувати однопоточному Node.js неблокуючі операції вводу-виводу, передаваючи їх виконування ядру системи у можливий момент.

Ініціалізація event loop відбувається в момент запуску сервера Node js, і з цього моменту він починає свою роботу, яку можна розділити на кілька етапів:

1. timers – виконання callback-функцій, зареєстрованих функціями setTimeout () і setInterval ()
2. pending callbacks – виклик callback-функцій операцій введення / виводу, виконання яких було відкладено на попередній стадії циклу подій
3. idle, prepare - виконання внутрішніх дій, необхідних самому Node js event loop
4. poll – виконання callback-функцій завершених асинхронних операцій і управління фазою timers
5. check – виконання callback-функцій, зареєстрованих функцією setImmediate ()
6. close callbacks – обробка раптово завершуються дій

У Node.js вам не потрібно хвилюватися про те, що відбувається на бекенда: досить просто використовувати коллбекі при роботі з INPUT/OUTPUT. Це дасть вам впевненість в тому, що ваш код не заблокує роботу сервера. На відміну від використання додаткових тредов або процесів, це хороше рішення з точки зору продуктивності.

Асинхронний INPUT/OUTPUT є перевагою, тому що операції введення / виведення набагато затретнее, ніж просте виконання коду[2]. До того ж, програма повинна займатися більш корисними справами, ніж постійно чекати INPUT/OUTPUT.

Подієвий цикл - це «сутність яка перехоплює і обробляє зовнішні події і конвертує їх у функції зворотного виклику». Тобто, виклики INPUT/OUTPUT - це певні точки, в яких Node.js може перемикатися від одного запиту до іншого. При зверненні до INPUT/OUTPUT, ваш код зберігає коллбек, і повертає контроль назад, в середу виконання Node.js. Збережено коллбек буде викликаний пізніше, коли всі необхідні дані будуть отримані.

Звичайно, на стороні бекенду, використовуються процеси і потоки для доступу до баз даних і для виконання різних процесів. Однак, все це не присутній явно в коді, так що не потрібно буде про це турбуватися. Єдине, про що варто пам'ятати, так це те, що операції INPUT/OUTPUT, наприклад, з базою даних або з іншими процесами будуть асинхронними з точки зору кожного запиту, і результати роботи цих процесів будуть повертатися через подієвий цикл назад в код. У порівнянні з моделлю роботи Apache, тут набагато менше потоків, як і витрачених на них ресурсів, тому що тут немає необхідності створювати новий тред для кожного запиту. Коли дійсно необхідно щось отримати або виконати щось в паралельному потоці, то управляти цим буде Node.js.

Node.js передбачає, швидку відповідь на вхідні запити, так що, такий підхід варто застосовувати не тільки до INPUT/OUTPUT: складні розрахунки, які завантажили процесор повинні бути винесені в окремі

процеси, з якими ви зможете взаємодіяти з допомогою подій. Або важкі операції можна винести з основного потоку, використовуючи такі абстракції, як WebWorkers. Це має на увазі, що ви не зможете распараллелить роботу вашого коду, без використання окремого фонового потоку, з яким ви зможете взаємодіяти з допомогою подій. В основному, всі об'єкти, які можуть кидати події (наприклад, інстанси EventEmitter) підтримують асинхронне взаємодія на основі коллбеков. Ви можете використовувати це в роботі з блокуючим кодом. Взаємодія можна організувати за допомогою файлів, сокетов, або дочірніх процесів, кожен з яких буде представлений в Node.js інстанси EventEmitter. Також можна реалізувати підтримку багатоядерності.

На Рисунку 4 зображена робота вишче описаного Event Loop.

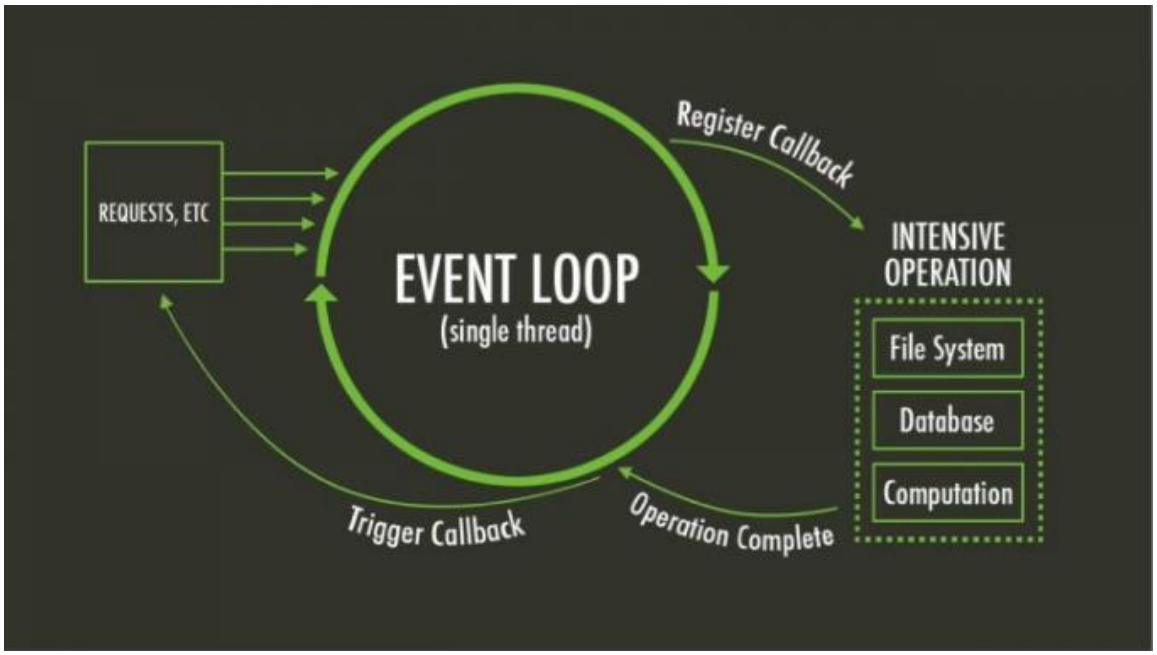


Рисунок 4 – як виглядає Event Loop

3.3. Express.js фреймворк

Це фреймворк, який дозволяє структурувати веб-додаток для обробки декількох різних запитів http за певною URL-адресою.

Express - це мінімальна, відкрита версія та гнучка рамка веб-додатків Node.js, призначена для того, щоб значно спростити розробку веб-сайтів, веб-додатків та API.

Фреймворк допомагає відповідати на запити з підтримкою маршруту, щоб користувач міг писати відповіді на конкретні URL-адреси. Підтримує декілька двигунів шаблонів для спрощення генерації HTML. Є дуже простим і має відкритий код.

Можливості Express.js:

Швидке програмування на стороні сервера. Будучи рамкою Node.js, Express.js пакує багато часто використовуваних функцій Node.js у функції, які можна легко викликати в будь-якій точці програми. Як наслідок, складні завдання, які в іншому випадку зайняли б розробник Node декількох сотень рядків і декількох годин, щоб запрограмувати програми, можуть легко виконати розробники Express JS лише за кілька рядків коду і протягом декількох хвилин. Тому розробка експрес-веб-додатків набагато швидша, ніж чиста розробка Node.js.

Маршрутизація. Маршрутизація дозволяє веб-додатку зберігати стани веб-сторінок за допомогою їх URL-адрес. Ці URL-адреси можуть бути надані іншим користувачам, і відвідування цих URL-адрес приведе користувачів до точного стану сторінки, яким було спочатку надано доступ. Node.js має механізм маршрутизації, але він є основним і рудиментарним. Express.js пропонує більш просунутий та ефективний механізм маршрутизації, який здатний обробляти високодинамічні URL-адреси.

Налагодження. Усі розробники стикаються з помилками в кожному проекті, що може призвести до несправності цілих додатків, і одне з найважливіших завдань розробників - виявити джерело цих помилок та виправити їх у найшвидший термін. Express.js забезпечує простий механізм налагодження, який дозволяє розробникам швидко визначити, яка частина програми викликає помилки.

Шаблон. Express.js забезпечує двигун шаблонів, який дозволяє веб-сторінкам мати динамічний контент, будуючи шаблони HTML на стороні сервера, замінюючи динамічний вміст їх належними значеннями, а потім

надсилає їх на сторону клієнта для візуалізації. На додаток до можливості динамічного контенту, він також сприймає значне навантаження з боку клієнта, що може мати сильно відрізняються технічні характеристики, і, як такий, він може зробити додатки більш ефективними.

Посереднє програмне забезпечення. Express.js використовує проміжне програмне забезпечення для систематичного впорядкування викликів різних функцій. Посереднє програмне забезпечення - це фрагмент або кластер коду, який має доступ до запиту користувача, відповіді програми та наступного середнього програмного забезпечення, яке слід використовувати. Завдяки такій архітектурі розробникам Express.js стає легко додавати, видаляти або змінювати різні функції до та з програми, що надає додаткові можливості високої масштабованості.

3.4. Мова програмування JavaScript

JavaScript - це мова програмування, що дозволяє реалізовувати складні функції на веб-сторінках - кожен раз, коли веб-сторінка робить більше, ніж просто сидіти там і відображати статичну інформацію, яку ви дивите - відображає своєчасні оновлення вмісту, інтерактивні карти, анімовані 2D / 3D-графіка, прокрутка відеороботів тощо - ви можете зробити ставку на те, що JavaScript, ймовірно, задіяний. Це третій шар з шарами стандартних веб-технологій, два з яких (HTML та CSS) ми висвітлювали набагато детальніше в інших частинах області навчання. Шари веб-технологій зображені на Рисунку 5, де: ARIA - Accessible Rich Internet Applications; CSS - Cascading Style Sheets, це мова правил стилю, яку ми використовуємо для застосування стилів до вашого вмісту HTML, наприклад, встановлення кольорів фону та шрифтів та викладення вашого вмісту в кілька стовпців; HTML - Hypertext Markup Language, це мова розмітки, яку ми використовуємо для структурування та надання змісту веб-контенту, наприклад, для визначення абзаців, заголовків та таблиць даних або вбудовування зображень та відео в сторінку.

Рисунок 5 - Шари стандартних веб-технологій

JavaScript - це технологія на стороні клієнта, вона в основному використовується для надання валідації на стороні клієнта, але вона має безліч функцій.

Можливості:

- JavaScript - це об'єктова мова сценаріїв.
- Надання користувачеві більшого контролю над браузером.
- Це обробка дат і часу.
- Він виявляє браузер користувача та ОС,
- Він має легку вагу.
- JavaScript - це сценарій мови, і це не Java.
- JavaScript - це сценарій мови інтерпретації.
- JavaScript враховує регістри.
- JavaScript - це об'єктна мова, оскільки вона забезпечує попередньо визначені об'єкти.
- Кожен оператор у javascript повинен закінчуватися крапкою з комою (;).
- Більшість синтаксисів керуючих операторів javascript відповідає синтаксису операторів керування на мові C.

- Важливою частиною JavaScript є можливість створення нових функцій в сценаріях. Оголосити функцію в JavaScript за допомогою ключового слова функції.

Обмеження JavaScript:

- Клієнтський JavaScript не дозволяє читати чи записувати файли.
- Його не можна використовувати для мережевих додатків, оскільки такої підтримки немає.
- Він не має ніяких багатопотокових або багатопроцесорних можливостей.

3.5. Mongoose JS модуль

Mongoose пропонує просто, на основі схеми рішення для моделювання даних вашої програми. Вона включає в себе вбудований тип лиття, валідацію, побудову запитів, гачки ділової логіки та інше. Має JSON структуру.

Mongoose - це картографічний об'єкт документа (ODM). Це означає, що Mongoose дозволяє визначати об'єкти із сильно набраною схемою, яка відображається на документі MongoDB.

Mongoose забезпечує неймовірну кількість функціональних можливостей навколо створення та роботи зі схемами. На даний момент Mongoose містить вісім схем типу, що властивість зберігається, як і коли воно зберігається в MongoDB.[3]

Типи схем:

- Рядок
- Номер
- Дата
- Буфер
- Булева
- Змішаний
- ObjectId

- Масив

Кожен тип даних дозволяє вказати:

- Значення за замовчуванням
- Спеціальна функція перевірки
- Вкажіть поле обов'язкове
- Функція get, яка дозволяє вам маніпулювати даними, перш ніж вони будуть повернуті як об'єкт
- Функція набору, яка дозволяє вам маніпулювати даними перед збереженням у базі даних
- Створити індекси, щоб дозволити швидше отримувати дані

Крім цих загальних параметрів, деякі типи даних дозволяють додатково налаштувати спосіб зберігання та отримання даних із бази даних. Наприклад, тип даних String також дозволяє вказати наступні додаткові параметри:

- Перетворити його в малі регістри
- Перетворити його в великі регістри
- Обрізання даних перед збереженням
- Регулярний вираз, який може обмежувати збереження даних, які можна зберегти під час процесу перевірки
- Перерахунок, який може визначити список рядків, які є дійсними

3.6. MongoDB

MongoDB - це міжплатформна програма, орієнтована на документи. MongoDB, класифікований як програма баз даних NoSQL, використовує документи, подібні JSON, зі схемою.

Можливості:

- Підтримка спеціальних запитів. У MongoDB можна здійснювати пошук за запитами по полях, діапазонах, а також підтримує регулярний пошук виразів.
- Індексація. Користувач можете проіндексувати будь-яке поле в документі.

- Реплікація. Майстер може виконувати читання та записи, а підлеглий копіює дані з ведучого і може використовуватися лише для читання або резервного копіювання (не записує)
- Копіювання даних. MongoDB може працювати на декількох серверах. Дані дублюються, щоб підтримувати систему, а також підтримувати її робочий стан у разі відмови обладнання.
- Балансування навантаження
- Підтримує засоби зменшення та агрегації карт.
- Використовує JavaScript замість процедур.
- Це база даних без схем, написана на C ++.
- Забезпечує високу продуктивність.
- Зберігає файли будь-якого розміру легко, не ускладнюючи ваш стек.
- Легкий у адмініструванні у разі відмов.

Він також підтримує:

- Модель даних JSON з динамічними схемами
- Автоматичне заточування для горизонтальної масштабованості
- Вбудована реплікація для високої доступності

3.7. React

Для розглядання цього фреймворка виділена велика частина роботи так як він основним у розробці панелі адміністратора та основоположні властивості React використовуються у React Native – що є мовою на якій написаний мобільний додаток.

React - це бібліотека javascript, яка була розроблена Facebook для користувальницького інтерфейсу в 2013 році. Він використовується для створення веб-сторінок або мобільних додатків на одній сторінці. Це потужна бібліотека, яка оптимальна для отримання даних, які швидко змінюються. Основна його функція - передача даних. DOM веб-сторінки - це те, на що спрямований ReactJS. Але найефективнішим є те, що він не оновлює всю веб-сторінку, натомість лише оновлює частину домену, яка

змінюється. Це значно підвищує продуктивність. Розберемо це на простому прикладі.

Усі ми використовуємо Facebook. Припустимо, один з наших друзів опублікував у Facebook фотографію, яку видно на нашій стрічці. Наразі на фотографії нашого друга немає нульових сподобань та коментарів. Коли ми натискаємо кнопку "подобається", кількість подібних піднімається вгору. Через кілька секунд кількість подібних та коментарів автоматично збільшується, оскільки інші люди також дають лайки та коментарі до фотографії нашого друга. Це все відбувається без перезавантаження сторінки.

Це відбувається через одну з особливостей ReactJS, відому як Virtual DOM. Є багато інших корисних функцій ReactJS.

3.7. 1Можливості і особливості React:

Існує багато відмінних функцій ReactJS, які дуже корисні при створенні інтерфейсу користувача.

1. Віртуальний DOM. Модель об'єкта документа або маніпуляція з DOM є однією з найважливіших частин Інтернету. Але оновлення DOM відбувається дуже повільно, навіть повільніше, ніж у багатьох операціях JavaScript. Більшість фреймворків JavaScript оновлюють весь DOM, що робить його повільніше. Не потрібно оновлювати весь DOM, натомість ці фреймворки повинні оновлювати лише ту частину DOM, яка необхідна для оновлення. Це те, що робить віртуальний DOM.

Віртуальний DOM - це легка копія оригіналу DOM. Він має всі властивості реального DOM. Коли відбувається оновлення, весь віртуальний DOM оновлюється. Це звучить неефективно, але віртуальний DOM набагато швидше, ніж оригінальний. Після завершення оновлення React порівнює оновлений віртуальний DOM і попередньо оновлений оригінальний DOM. Це відомо як процесс

Diffing. Порівнюючи обидва DOM, React точно знає, де різниця. Тож він оновлює оригінальний DOM лише там, де була різниця.

2. Одностороння прив'язка даних. Одностороння прив'язка даних означає, що через всю програму дані протікають лише в одному напрямку. Це забезпечує кращий контроль над ним. Дані передаються від батьківського компонента до дочірнього компонента через реквізити для читання. Ці реквізити не можна повернути до батьківського компонента. Так працює одностороння прив'язка даних. Хоча дочірній компонент може спілкуватися з батьківським компонентом для оновлення стану за допомогою функцій зворотного дзвінка.

3. Компоненти. Веб-сторінка в React розділена на різні компоненти. Кожен компонент визначає подання або його частину. React заснований на компонентах. Логіка компонентів написана в JavaScript, а не в шаблонах, тому легко передавати дані через додаток і не підтримувати стан DOM.

4. JSX. JSX - це розширення синтаксису javascript. Його синтаксис схожий на HTML. Компоненти ReactJS записані в JSX. JSX можна розглядати як комбінацію javascript та XML. Його синтаксис дуже простий, що робить написання компонентів дуже простим. Про JSX ми поговоримо пізніше.

5. Динамічні Стани. Однією з найкращих особливостей ReactJS є те, що ми можемо використовувати стани компоненту всередині JSX. Це дуже допомагає під час відображення даних у браузері відповідно до умов.

6. Методи життєвого циклу. Методи життєвого циклу, які надає ReactJS, дуже корисні при розробці. Кожен компонент ReactJS має власний життєвий цикл. Вони являють собою низку методів, які виконуються на різних етапах виконання. Перелік методів життєвого циклу у порядку від створення компоненту до його знищення:

1. shouldComponentUpdate
2. componentDidMount
3. componentWillUnmount
4. componentDidUpdate

3.7.2 JSX. JSX - одна з особливостей ReactJS, яка допомагає в створенні елементів ReactJS дуже ефективно. Це розширення Javascript. JSX включає як логіку, так і розмітку. На відміну від AngularJS, нам не потрібно створювати окремі файли для логіки та розмітки. Це економить багато часу.

Наступний код написаний в JSX.

var element = <h1> This is JSX </h1>

У наведеному вище коді ми зберігаємо тег заголовка HTML всередині змінної. Є багато переваг використання JSX:

– JSX швидше, ніж Javascript.
– Логіку та розмітку можна записати всередині одного файлу.
– Створити шаблони в JSX легко.

Розберемося більше про JSX на прикладі.

```
function App() {
  var element = <h1> This is JSX </h1>
  return(element)
}
```

Дотримуйтесь наведеного вище коду. Існує функція, названа як додаток. Ця функція виглядає як звичайна функція javascript. У другому рядку ми використовували змінну, елемент і дали їй заголовок HTML як значення. Це поєднання HTML та Javascript. Він дуже поширений у JSX. У третьому рядку ми повернули створену нами змінну. Так працює JSX. Замість повернення єдиної змінної ми також можемо повернути HTML-код.

3.7.3 Переваги React:

– ReactJS використовує віртуальний DOM, що покращує роботу користувачів. Це також робить роботу розробника менш складною.

– JSX використовується в ReactJS, який дуже простий і легкий у навчанні.
– У ReactJS немає необхідності в окремих файлах для логіки та розмітки.
– ReactJS - це бібліотека з відкритим кодом, яка підтримується Facebook. Це постійно розвивається бібліотека.
– Прив'язка до однієї інформації робить код дуже стабільним.
– ReactJS також пропонує мобільне рішення, відоме як React Native.
– Виконує re-render швидше ніж фрейворки-конкуренти.
– Умовні висловлювання в ReactJS дуже корисні.
– Фреймворк має можливість робити SEO оптимізацію.
– Він має чудовий набір інструментів для розробників.

3.7.4 Мінуси ReactJS:

– Потрібні додаткові бібліотеки для маршрутизації, управління державою та взаємодії API.
– ReactJS - це бібліотека великих розмірів.
– Темп розвитку дуже високий.
– ReactJS охоплює лише частину інтерфейсу користувача, більше нічого.
– Через дуже високий темп розвитку документація ReactJS погано зберігається. [4]

3.8 React Native

React Native - це фреймворк для мобільних додатків з відкритим кодом, створена Facebook. Він використовується для розробки додатків для Android, iOS, Web та UWP, дозволяючи розробникам використовувати React разом із можливостями рідної платформи React.

Можливості:

Write Once and Use Everywhere. На сьогоднішній день найкраща особливість та реальна концепція, на чому ґрунтується код, написаний у

роботі React Native майже на кожній мобільній платформі. Він включає IOS, Android, Windows тощо. Нам не потрібно писати код у Swift для IOS, java для android або C # для Windows. Програми, створені за допомогою React native, є рідними, це означає, що вони працюють на декількох платформах

Мова програмування рамки - одна важлива частина, яка дуже впливає на розробників. Якщо рамка використовує популярну і широко використовувану мову програмування, вона завжди впливає на вибір розробника. React Native використовує одну таку мову, Javascript. Без сумніву, javascript - одна з найпопулярніших і широко використовуваних мов програмування. Це одна з трьох основних технологій Всесвітньої павутини (WWW), а інші дві - HTML та CSS. Будь-яка людина, яка має навіть невеликий досвід веб-розробки, знає, як використовувати JavaScript. Крім того, популярні веб-рамки також побудовані на JavaScript. Це також просто і легко вивчити мову програмування.

Користувальницький інтерфейс зосереджений. React Native суто орієнтований на дизайн інтерфейсу. Він чудово реагує, і його здатності до рендерингу є на сьогодні найкращими.

Час розробки. Час розвитку в React Native значно короткий.

Підтримка сторонніх бібліотек. Використання сторонніх бібліотек завжди є плюсом. Це дає свободу вибору розробникам. Це одночасно добре і погано.

NPM для встановлення. Процес установки - це завжди головний біль, особливо для новачків. React Native використовує для встановлення Node Package Manager або NPM, і використовувати його досить просто. Люди, які не мають JavaScript, мають досвід роботи з NPM, а також новачкам не так складно вивчати команди NPM.

Продуктивність мобільних середовищ. Більшість нативних програм орієнтовані на процесор, тоді як програми, побудовані за допомогою React

Native, орієнтовані на GPU. Це призводить до кращої продуктивності, ніж орієнтовані на CPU програми.

Live Reload. Функція перезарядження в реальному часі забезпечує два екрани: один для зміни коду та інший для перегляду модифікації. Кожен, хто має досвід розробки мобільних додатків, знає, як ця функція може змінити ситуацію.

Мова програмування. Як обговорювалося раніше, одним із плюсів використання React Native є мова, якою він користується. Програми в React Native будуються за допомогою JavaScript. Аналогічно Ionic також використовує Javascript разом із HTML та CSS. Але Flutter трохи інший. Мова програмування, яка використовується у Flutter, - це Dart. Dart був представлений Google в 2011 році і не настільки популярний, як Javascript, навіть не близько.

Потоки у React Native:

- Потік інтерфейсу користувача: Також відома як головна нитка. Це використовується для рідного рендеринга інтерфейсу Android та iOS. Наприклад, в android цей потік використовується для вимірювання / компонування / малювання.
- Потік JS: потік JS або потік Javascript - це потік, де буде працювати логіка. Наприклад, це потік, в якому виконується код javascript програми, здійснюються запити до API, обробляються події дотику. Оновлення до власних представлень збираються та надсилаються на рідну сторону в кінці кожного циклу події в потоці JS. Щоб підтримувати хороші показники, потік JS повинен мати можливість надсилати пакетні оновлення до потоку інтерфейсу до наступного строку візуалізації кадру. -
- Рідний модульний потік: Іноді додатку потрібен доступ до API платформи, і це відбувається як частина рідного модульного потоку.

- Візуалізація потоку: Тільки в Android L (5.0) реактивна нитка візуалізації використовується для створення фактичних команд OpenGL, які використовуються для малювання вашого інтерфейсу користувача.

Процеси у роботі React Native:

1. При першому запуску програми головний потік починає виконання і починає завантажувати пакети JS.
2. Коли код JavaScript успішно завантажений, головний потік надсилає його в інший потік JS, оскільки коли JS виконує деякі важкі обчислення, наповнюють нитку на деякий час, потік інтерфейсу користувача не буде страждати жодного разу.
3. Коли React почне рендеринг, Reconciler починає «відрізнятися», а коли генерує новий віртуальний DOM (макет), він надсилає зміни до іншого потоку (Shadow thread).
4. Shadow thread обчислює макет, а потім відправляє параметри / об'єкти макета в основний потік (UI). (Тут ви можете запитати, чому ми називаємо це «тінь»? Це тому, що він генерує тіньові вузли)
5. Оскільки лише основний потік здатний щось відобразити на екрані, тіньовий потік повинен надсилати сформований макет до основного потоку, і лише потім відображається UI.

Основні 3 частини React Native зображені на Рисунку 6:

- React Native - Рідна сторона
- React Native - JS сторона
- Міст

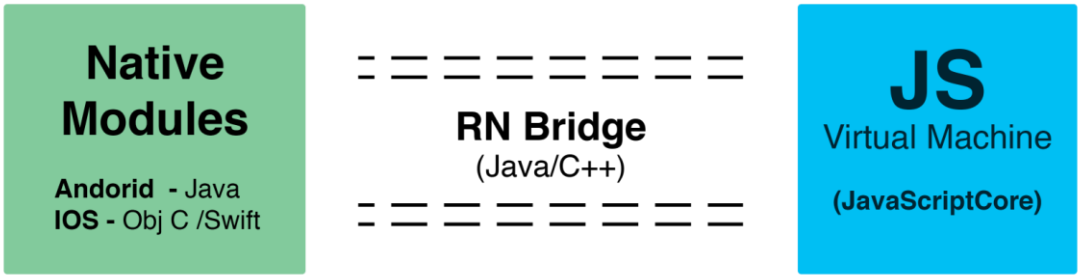


Рисунок 6 – основні частини React Native додатку

3.9. Git

Git - це безкоштовна та з відкритим кодом розповсюджена система управління версіями, призначена для швидкого та ефективного управління всіма проектами від маленьких до дуже великих проектів.

Git легко засвоїти та має крихітний слід із блискавичною швидкістю. Він випереджає такі засоби SCM, як Subversion, CVS, Perforce та ClearCase з такими функціями, як дешеве місцеве розгалуження, зручні місця постановки та безліч робочих процесів.

Під час розробки на платформу контролю версій github.com було завантажено 5 версій.

3.10 Тайм-трекер Clockify

Для контролю часу під час написання дипломної роботи було використано тайм-трекер Clockify. Цей інструмен дозволяє рахувати, аналізувати витрачений час на первні задачі. На основі даних с цієї системи було проаналізовано на які задачі було витрачено найбільше часу. На рисунку 7 зображено приклади маркування задач та витрачений час на них. Як можно побачити, то деякі задачі вимагали повторної доробки. На рисунку 8 зображено кількість годин витрачену на написання дипломного звіту та доробки невеликих частин програмного продукту.

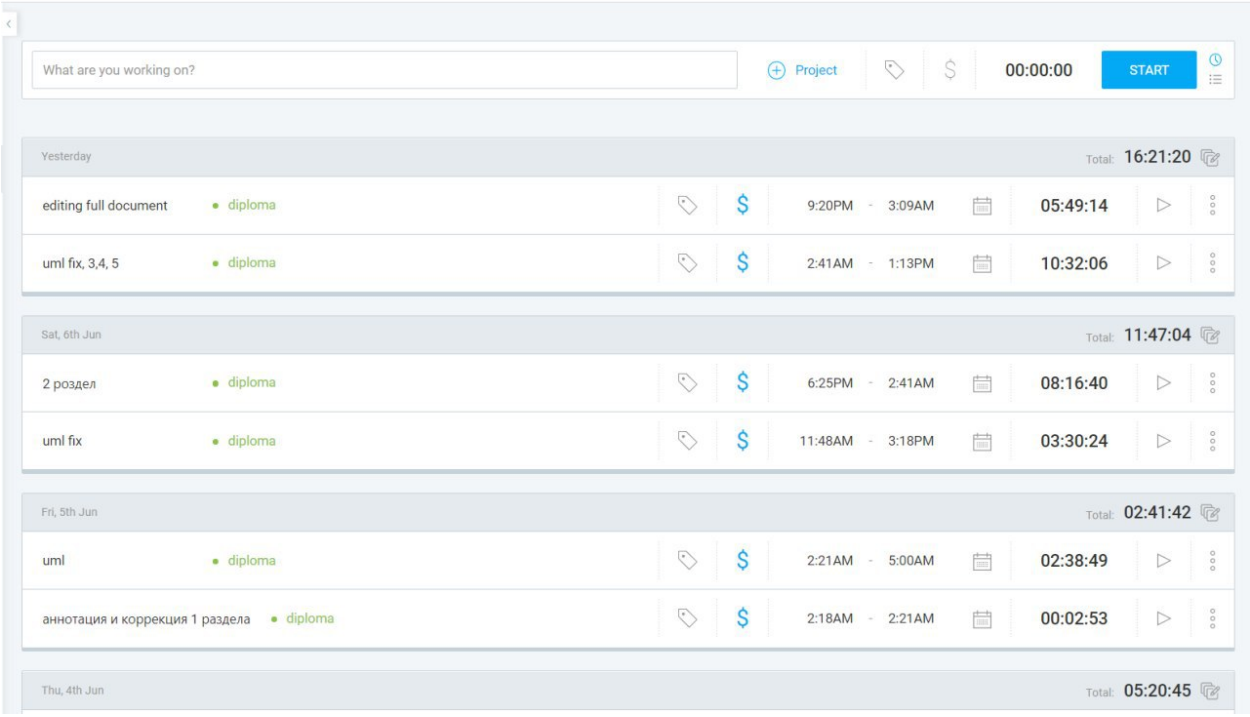


Рисунок 7

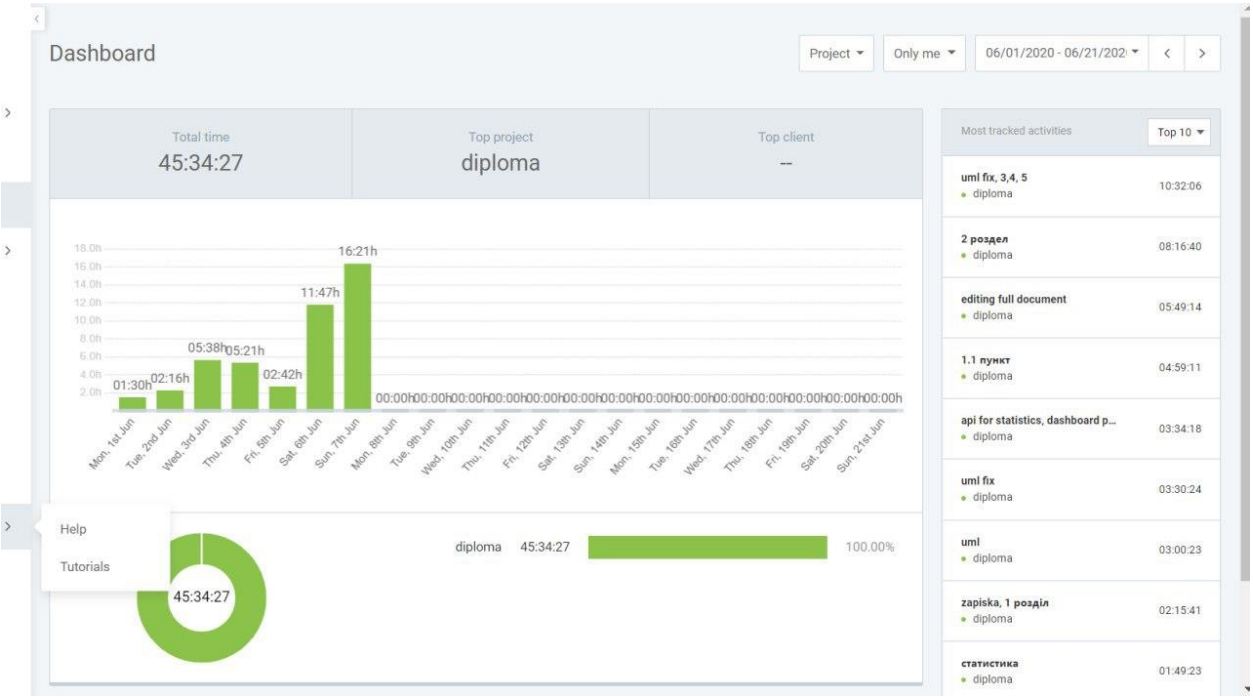


Рисунок 8

3.11 ВИСНОВКИ ДО 3 РОЗДІЛУ

У цьому розділо було визначено стак технологій, що буде використовуватися для розробки системи. MERN – цей стак буде використаний, так як він побудований на одній мові Javascript, що дає перевагу у швидкості розробки так як розробнику не треба переходити на іншу у, наприклад, ситуації: розробка backend на одній мові, а frontend – на інший.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ

Кожен з елементів системи є запущеним сервером на різних портах локальної мережі:

- Порт 3000 – веб-сервер;
- Порт 4000 – панель адміністратора;
- Порт 8081 – відладка та встановлення мобільного додатку на пристрій з якого проходить тестування;

Під час розробки і тестування продукту кожен сервер запущенний локально на комп'ютері, де проводиться розробка. Кожен сервер доступний за посиланням http://localhost:3000 або http://localhost:4000, та http://localhost:8081, для кожної частини проекту. Тобто усі запити між компонетами системи йдуть по локальній мережі окрім запитів до бази даних, так як вона розташована «хмарно».

Зв'язок між серверами під час розробки було здійснено за допомогою методу proxy у файлу package.json. Веб-сервер панелі адміністратора має у конфігураціонному файлі проксування до backend серверу, а на backend сервері – відповідно навпаки. Цей метод взаємодії з сервером має назву «прямі проксі»[6].

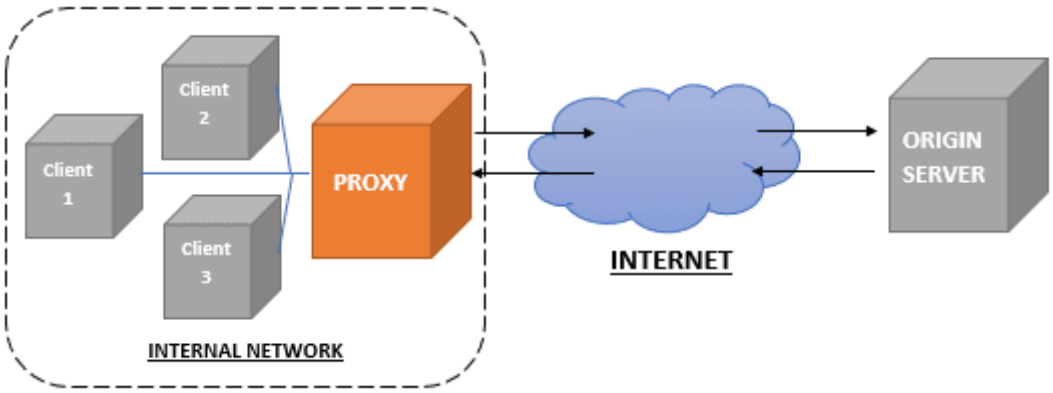


Рисунок 1

На рисунку 1 зображено то, як працюють проксі на сервері. Як вони працють: при запиті з клієнту 1, йде запит на первний роут, котрого немає на одному

сервері, то цей запит з цим роутом йде автоматично на сервер, що указаний у як проксі. Тобто це є елементом кліент-серверної архітектури.

При розробці проксі запити йшли на два порти: http://localhost:4000, http://localhost:3000 від одного до іншого.

4.1 Реалізація серверної частини.

Як було описано у попередньому розділі, серверна частина написана на Javascript, основним фреймворком для розробки є Express JS і запуск та робота серверу виконується за допомогою Node JS.

4.1 Архітектура серверу

Детальне зображення веб-сверу ієрархії файлів зображено на Рисунку 1.

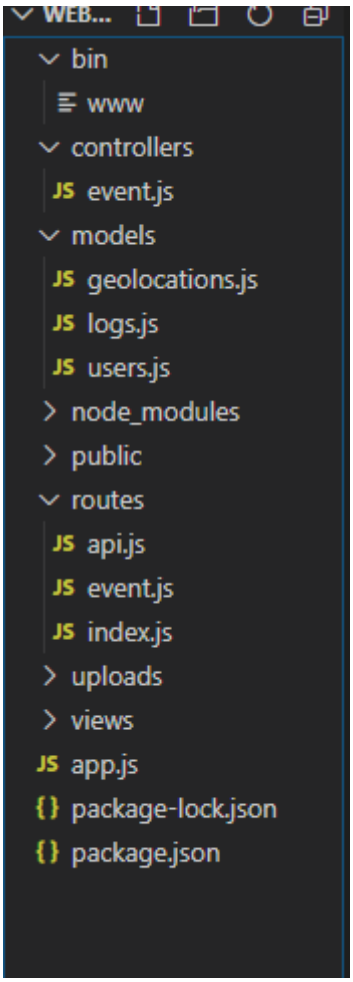


Рисунок 1.

Опис файлів з Рисунку 1:

App.js основний виконуючий файл, містить у собі настройки серверу, код

підключення бази даних та основні настройки маршрутизації запитів, що будуть надходити до северу.

Маршрутизація зсилається на кінцеві точки програми (URI) реагують на HTTP запити клієнтів.

4.1.1 Основні види HTTP запитів:

- GET Метод GET вимагає представлення вказаного ресурсу. Запити, що використовують GET, повинні отримувати лише дані.
- HEAD. Запитує заголовки, які повертаються, якщо вказаний ресурс буде запрошено методом HTTP GET. Такий запит можна зробити перед тим, як вирішити завантажити великий ресурс, щоб зберегти пропускну здатність, наприклад.
- POST Метод POST використовується для подання об'єкта на вказаний ресурс, часто викликаючи зміну стану або побічні ефекти на сервері.
- PUT. Метод запиту HTTP PUT створює новий ресурс або замінює представлення цільового ресурсу на корисне навантаження запиту.
- DELETE Метод DELETE видаляє вказаний ресурс.
- CONNECT Метод CONNECT встановлює тунель до сервера, ідентифікований цільовим ресурсом.
- OPTIONS Метод OPTIONS використовується для опису параметрів зв'язку для цільового ресурсу.
- TRACE Метод TRACE виконує тест зворотного зв’язку повідомлення по шляху до цільового ресурсу.
- PATCH Метод PATCH використовується для застосування часткових модифікацій до ресурсу.

Методи маршрутизації у Express JS задають функцію зворотного виклику іноді її називають "функцією обробника", яка викликається, коли програма отримує запит на вказаний маршрут (кінцеву точку) та метод HTTP. Іншими словами, програма «слухає» запити, які відповідають вказаному маршруту

(маршрутам) та методу (ім), і коли виявляє відповідність, викликає вказану функцію зворотного виклику.

4.1.2 Паттерн MVC

На рисунку 1 зображено три папки з назвами «model», «view», «controller», що мають у середині виконуючі файли. Файли з цих папок є частинами паттерну програмування з назвою MVC.

Model-View-Controller (або MVC) - це, мабуть, одна з найпопулярніших архітектур для додатків. Як і багато інших цікавих речей в історії комп'ютерів, модель MVC була задумана в PARC для мови Smalltalk як рішення проблеми організації програм із графічними інтерфейсами користувачів. Він був створений для настільних додатків, але відтоді ідея була адаптована до інших середовищ, включаючи Інтернет.[8]

Ми можемо описати архітектуру MVC простими словами:

- Model: частина нашого додатку, яка буде мати справу з базою даних або будь-якими функціональними даними.
- View: все, що побачить користувач - в основному це сторінки, які ми збираємося надсилати клієнту.
- Controller: логіка нашого сайту та клей між моделями та видами. Тут ми закликаємо наші моделі отримувати дані, а потім ми ставимо ці дані на наші погляди, щоб надсилати користувачам.

Наш додаток дозволить нам створювати, переглядати, редагувати та видаляти текстові нотатки. Він не матиме інших функціональних можливостей, але оскільки у нас вже буде чітка архітектура, ми не будемо мати багато проблем із масштабуванням проекту.

Відповідно до теорії було реалізовано взаємодії з базою даних за допомогою трьох моделей: Logs, Geolocations, Tags. відповідно до колекцій у базі дани. Основна ціль моделі - це встановлення назви полів, їх типу даних та я яку саме колекцію буде здійснено запис.

У ролі View є додатковий React сервер, так як він має ширший фунціонал, ніж вбудовані фреймворки для створення інтерфейсу користувача. Контроллери містять у собі алгортими, що будуть взаємодіяти з базою даних, основна їх ціль – формувати, перевіряти дані для подальшого використання їх моделями.

4.1.3 Впровадження хмарної бази даних MongoDB

Переваги використання хмарних баз даних замість звичайних локальних:

– Легкість доступу. Користувачі можуть отримати доступ до баз даних хмари практично з будь-якого місця, використовуючи API постачальника або веб-інтерфейс.
– Масштабованість. Хмарні бази даних можуть розширювати свої сховища під час роботи, щоб задовольнити зміни потреб. Організації платять лише за те, що вони використовують.
– Аварійного відновлення. У разі стихійного лиха, відмови обладнання або відключення електроенергії, дані зберігаються в безпеці за допомогою резервних копій на віддалених серверах[7].

На рисунку 2 відображено веб-панель хмарної бази даних.

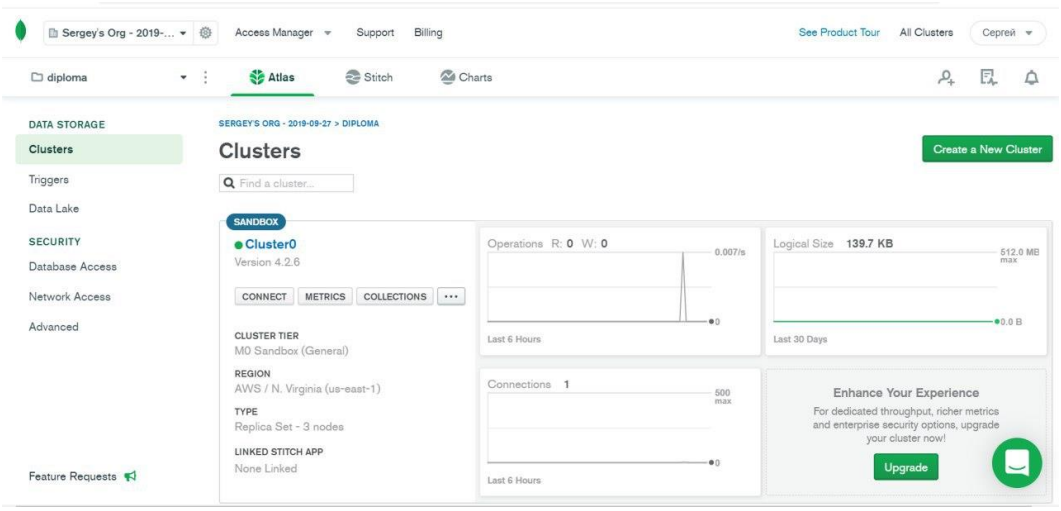


Рисунок 2

4.2 Реалізація мобільного додатку. Мобільний зчитувач карт-пропусків

Цю частину системи було реалізовано за допомогою фреймворку React Native. Основна функція мобільного зчитувачу – сканувати дані з NFC міток, відправляти дані на веб-сервер та отримувати відповідь для проскановаої мітки.

Можливі відпові: користувач має право доступу, користувач немає права доступу.

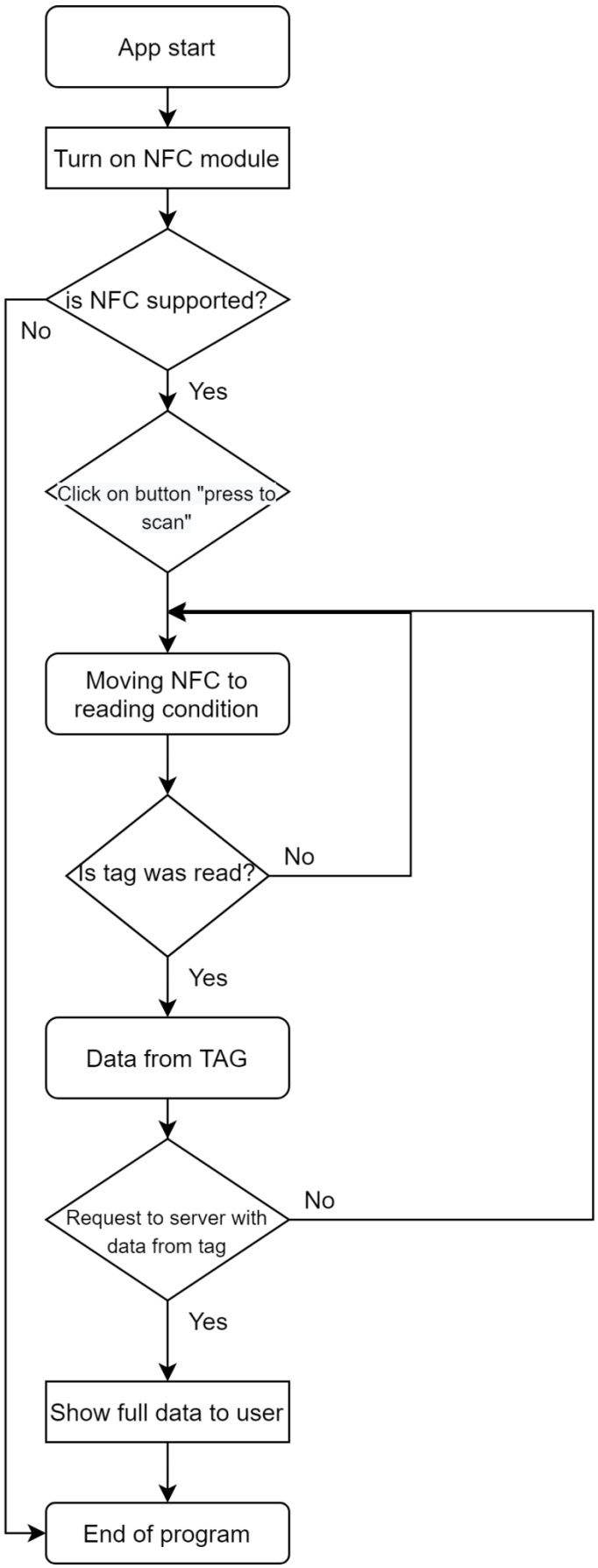

App start
Turn on NFC module
is NFC supported?
No
Yes
Click on button "press to scan"
Moving NFC to reading condition
Is tag was read?
No
Yes
Data from TAG
Request to server with data from tag
No
Yes
Show full data to user
End of program

Рисунок 3

На рисунку 3(три) зображено алгоритм роботи програми.

Детальний опис роботи програми:

Під час запуску додатку програма вмикає модуль NFC для перевірки його наявності на пристрої, у випадку відстутності, то буде видана помилка користувачеві. Далі головний екран програма очікує натискання на кнопку сканування, після натискання модуль NFC починає сканувати мітки, у цей час користувач повинен прикласти NFC мітку. На цьому етапі може бути 2 результати: помилка зчитування мітки – у цьому разі сканування потрібно починати спочатку, інший варіант – успішне сканування з отриманням даних з мітки. Наступним етапом є відправлення даних з мітки, інформації про мітку до веб-серверу. Результатом запиту до веб-серверу є відповідь з повною інформацією з мітки.

4.2.1 Опис додаткового фунціоналу

Додаток має другий екран по свайпу у ліву сторону. Розділ програми призначений для розробників. Основна функція: перезаписувати, записувати дані на мітки. Цей екран містить поле вводу, та кнопку щоб почати запис.

4.2.3 Опис протоколу NDEF

У цьому розділі описується протокол через який йде взаємодія між картками-пропусками та смартфоном с додатком.

Формат обміну даними NFC (NDEF) - це стандартизований формат даних, який може використовуватися для обміну інформацією між будь-яким сумісним пристроєм NFC та іншим пристроєм або тегом NFC. Формат даних складається з повідомлень NDEF та записів NDEF. Цей стандарт підтримується Форумом NFC і є вільним для консультацій, але для його завантаження потрібно прийняти ліцензійну угоду.

Формат NDEF використовується для зберігання та обміну інформацією, наприклад URI, звичайним текстом тощо, використовуючи загальновідомий формат. Теги NFC, такі як карти Mifare Classic, можна конфігурувати як теги NDEF, а дані, записані до них одним пристроєм NFC

(Записи NDEF), можуть бути зрозумілі та доступні будь-яким іншим сумісним NDEF пристроєм. Повідомлення NDEF також можуть використовуватися для обміну даними між двома активними пристроями NFC в режимі "одноранговий". Дотримуючись формату обміну даними NDEF під час спілкування, пристрої, які в іншому випадку не мали б значущих знань один про одного або загальної мови, здатні обмінюватися даними впорядкованим, взаєморозумілим чином.

4.3 Реалізація панелі адміністратора.

Панель адміністратора є працюючим React сервером. Додаток складається компонетнів. Дані у компоненти поступають за допомогою методу GET HTTP у форматі JSON. Ці дані зберігаються у локальному стані компоненту. Далі ці дані виводяться у зручному та зрозумілому для користувача вигляді. Для зміни чи оновлення даних у базі даних використовувався POST HTTP запит.

Нижче на рисунку 4 зображено структуру серверу.

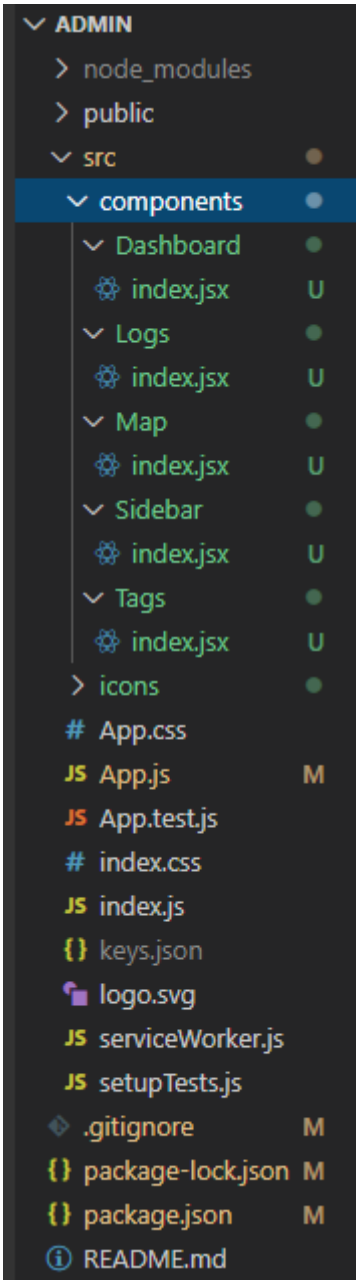


Рисунок 4

На цьому рисунку зображено:

Index.js – файл, що поміщає усі компоненти серверу у Virtual DOM;

App.js – основний виконавчий файл, містить у собі підключення модулю для роутінгу та історії переміщень по сторінкам, містить сам роунтінг компонентів та статичний елемент лівого меню – зображено під номером 1 у лівій частині на рисунку 5.

Папка components – містить у собі папки с виконуючими файлами компонентів системи, кожен з компонентів буде відображений на рисунку 5 у області під номером 2.

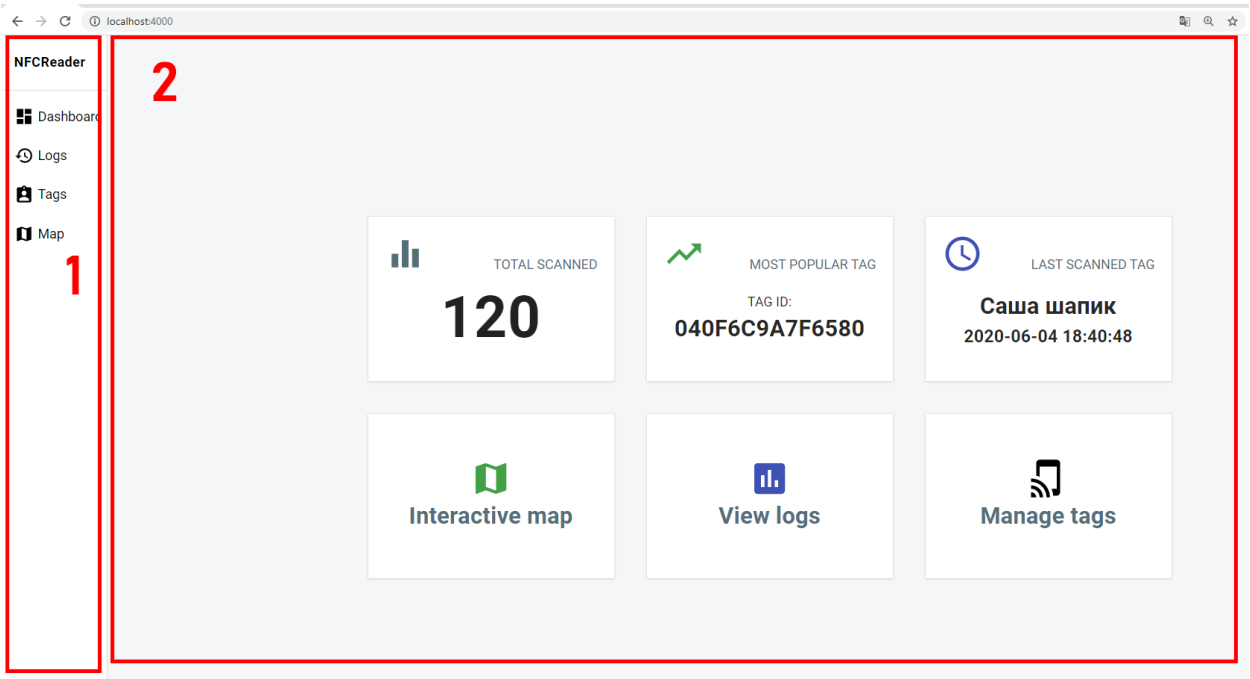


Рисунок 5

4.4 ВИСНОВКИ РОЗДІЛУ 4

У цьому розділі було сформовано структуру кожного елементу системи, встановлено метод взаємодії компонентів системи, а саме була використана кліент-серверна архітектура. Буда впроваджена хмарна база даних. Описано паттерни проектування, що будуть використані у побудові додатків системи. Розглянено питання взаємодії серверів між собою.

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Взаємодія з системою проходить у двох видах: взаємодія с веб-додатком панелі адміністратора та взаємодія с мобільним додатком, що є зчитувачем міток.

5.1 Взаємодія з мобільним додатком

Мобільний додаток має 2 стартові екрани:

Стартовий екран для сканування зображений на рисунку 1. Для сканування потрібно нажати на кнопку «Press to scan» та прикласти NFC мітку. Далі після зчитування даних з мітки буде відправлено запит на сервер, під час цього юзер буде бачити вікно загрузки – Рисунок 3. Після успішного утримання усіх даних інформація про користувача мітки буде зображена на екрані - рисунок 4.

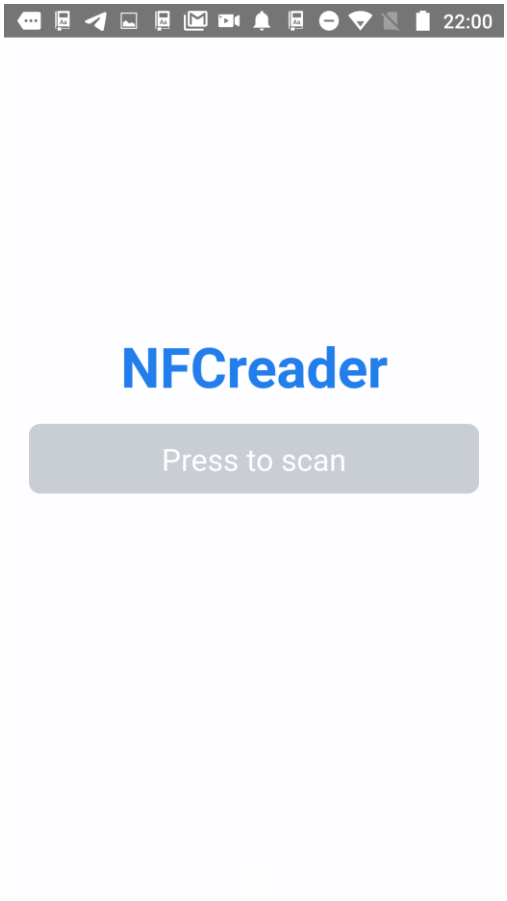


Рисунок 1

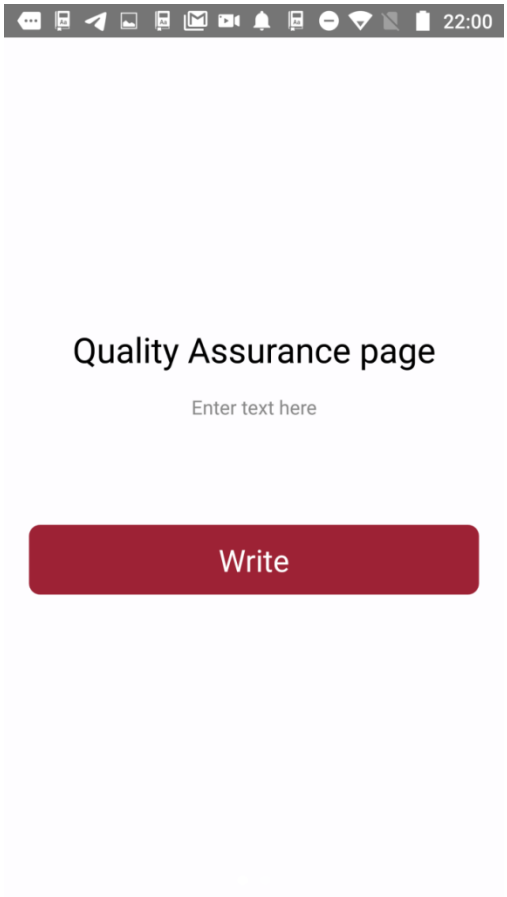
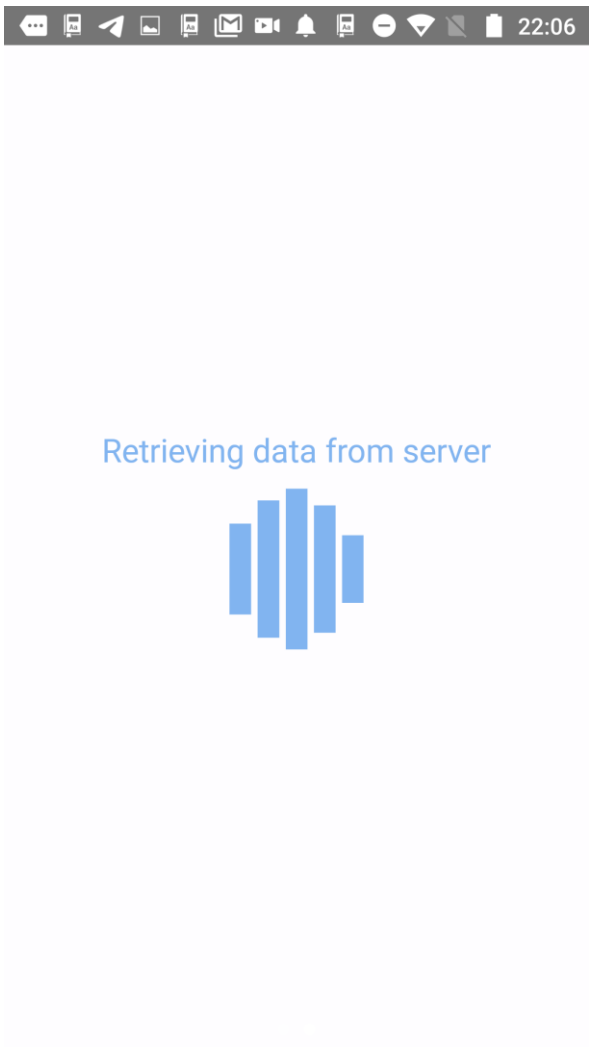


Рисунок 2



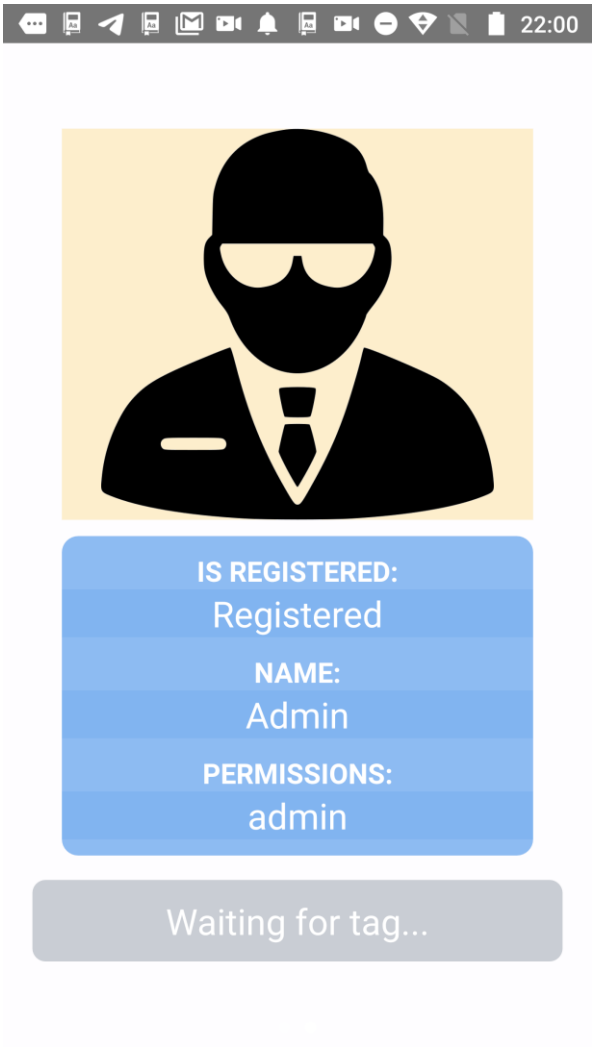


Рисунок 3 *Рисунок 4*

Дані, що відображаються на екрані успішного відсканування:

1. Фотокартка користувача
2. Статус реєстрації користувача у системі
3. Ім’я
4. Права доступу

На основі цих даних користувач додатку вирішує пропускати або не пропускати користувача картки на приватну територію.

Можливі варіації результату сканування:

Користувач не зареєстрований у системі, приклад – рисунок 5

– Користувач зареєстрований, але не має прав доступу, приклад – рисунок 6
– Користувач має права доступу, приклад – рисунок 4

Можливі права користувачів:

– Admin
– Worker
– None

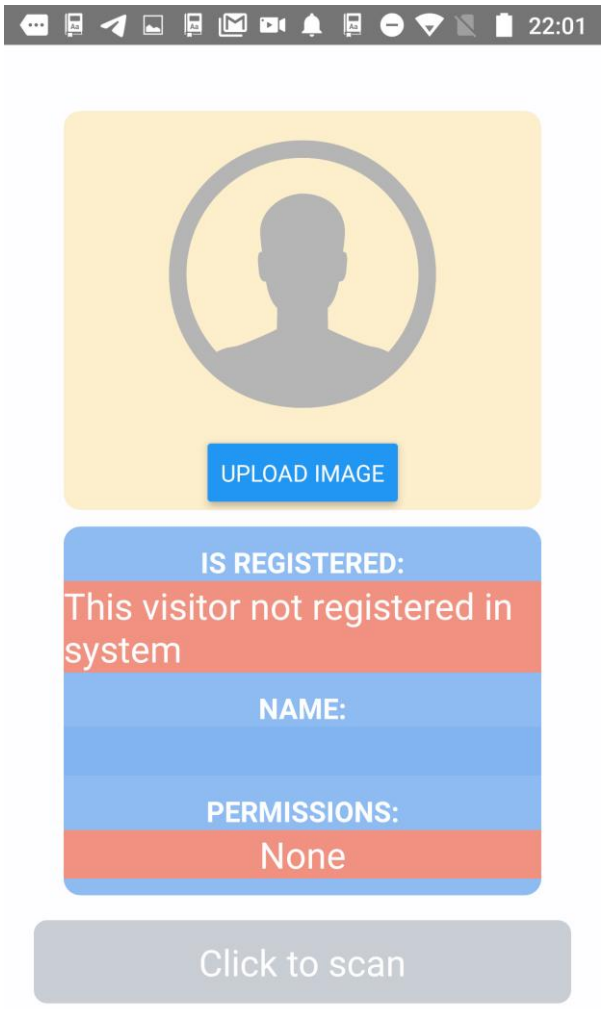


Рисунок 5

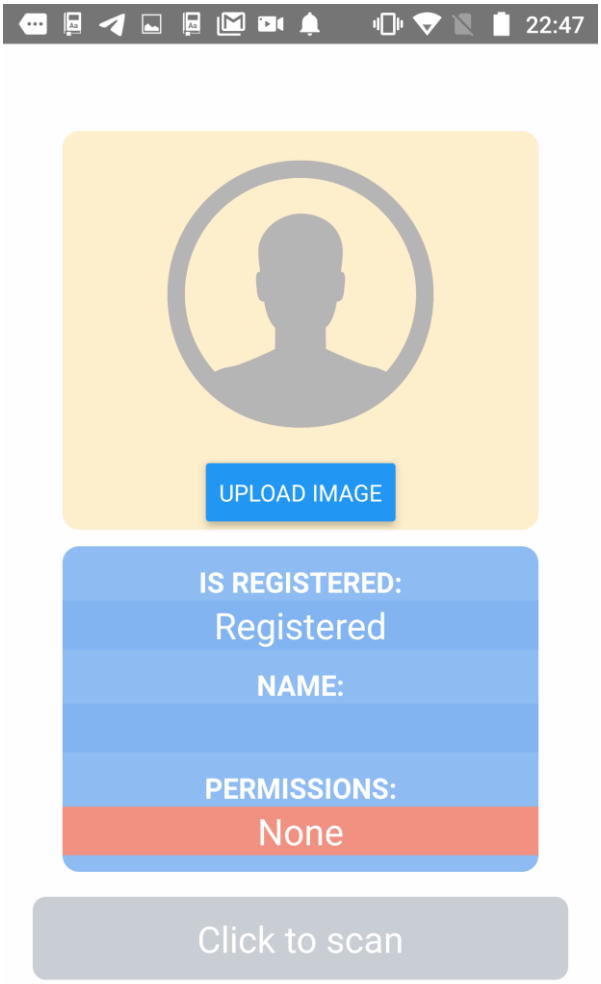


Рисунок 6

5.1.2 Огляд додаткових опцій

Додатковою опцією є можливість додати фото, для цього треба нажати на кнопку «upload image», рисунок 7 та 8 відображає цей процес. Фото можна обрати з галереї телефону або зробити новий знімок, після встановлення

фотографії, користувеві буде повідомлено про успішну загруку фото, також воно буде відображено у картці користувача.

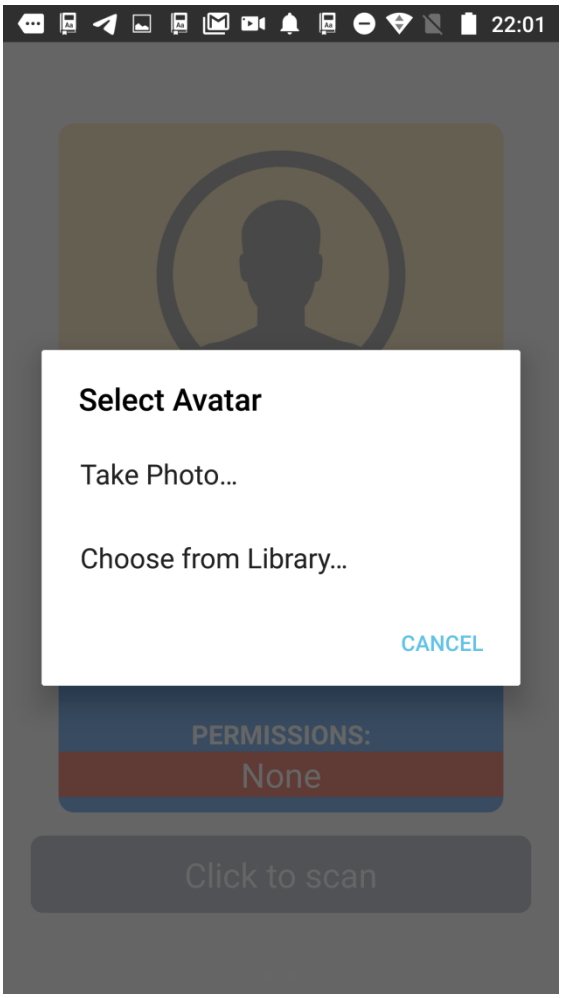


Рисунок 7

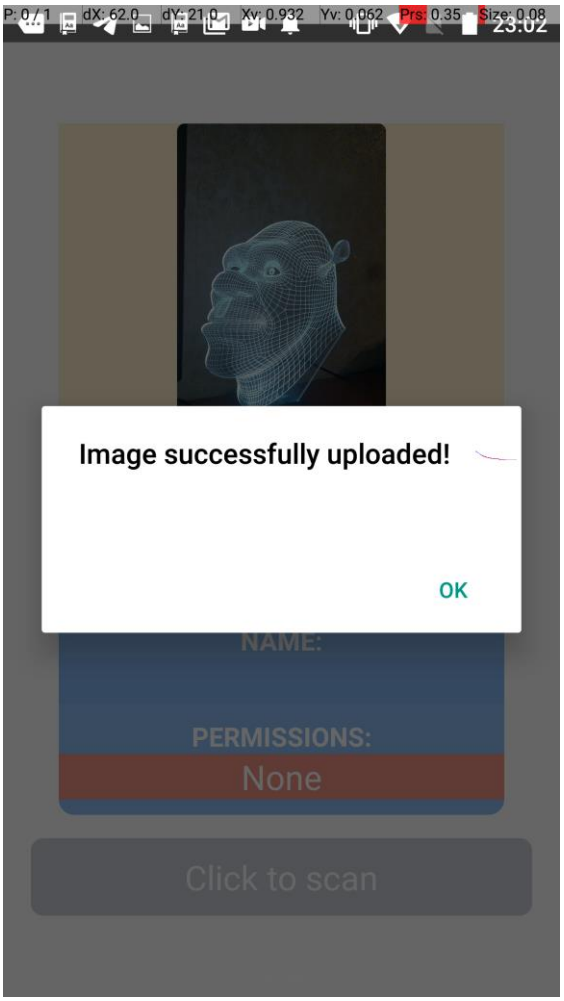


Рисунок 8

Такой на рисунку 2 відображена функція для розробників, що дає можливість змінювати дані, що містяться на мітці.

5.2 Взаємодія з панеллю адміністратора

Панель адміністратора є веб-ресурсом, що доступний за посиланням http://localhost:4000/ при запуску серверу локально.

Перша сторінка панелі є міні Dashboard-ом, зображення – рисунок 9

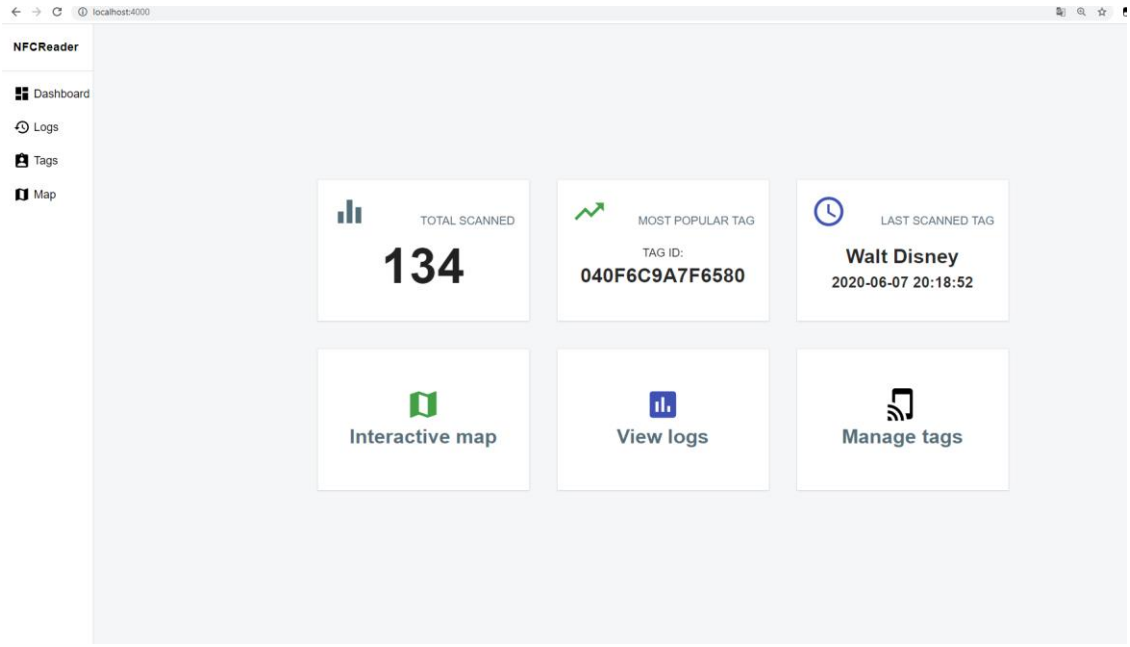


Рисунок 9

Зображення має:

– Навігаційне меню зліва
– Невелику статистику трьох видів у центрі екрана, перший рядок із трьох блоків. Другий рядок – також навігаційні кнопки, що ведуть до інших розділів.

Зміст навігаційного меню:

– Dashboard – зображено на рисунку 9
– Logs – сторінка з журналом подій
– Tags – сторінка що містить детальну інформацію усіх міток системи
– Map – інтерактивна мапа, що відображає місця сканувань та інформацію про сканування.

5.2.1 Вкладка журналу подій

Журнал подій відображає події за наступними властивостями:

– Event Type – тип події, можливі типи подій:
    - Scan – сканування мітки
    - Edit – зміна даних профілю з панелі адміністратора
    - Photoupload – оновлення фото контролером з мобільного додатку

- registertag – реєстрація нової мітки у системі

– Identificator NFC tag – унікальний ідентифікатор кожної NFC мітки
– Date – поточна дата події
– Author - ідентифікатор прострою згенерований на мобільному додатку або помітка «admin», якщо подія була с панелі адміністратора
– IP – IPv4 адреса пристрою з якогу була подія

Журнал подій зображений на рисунку 10

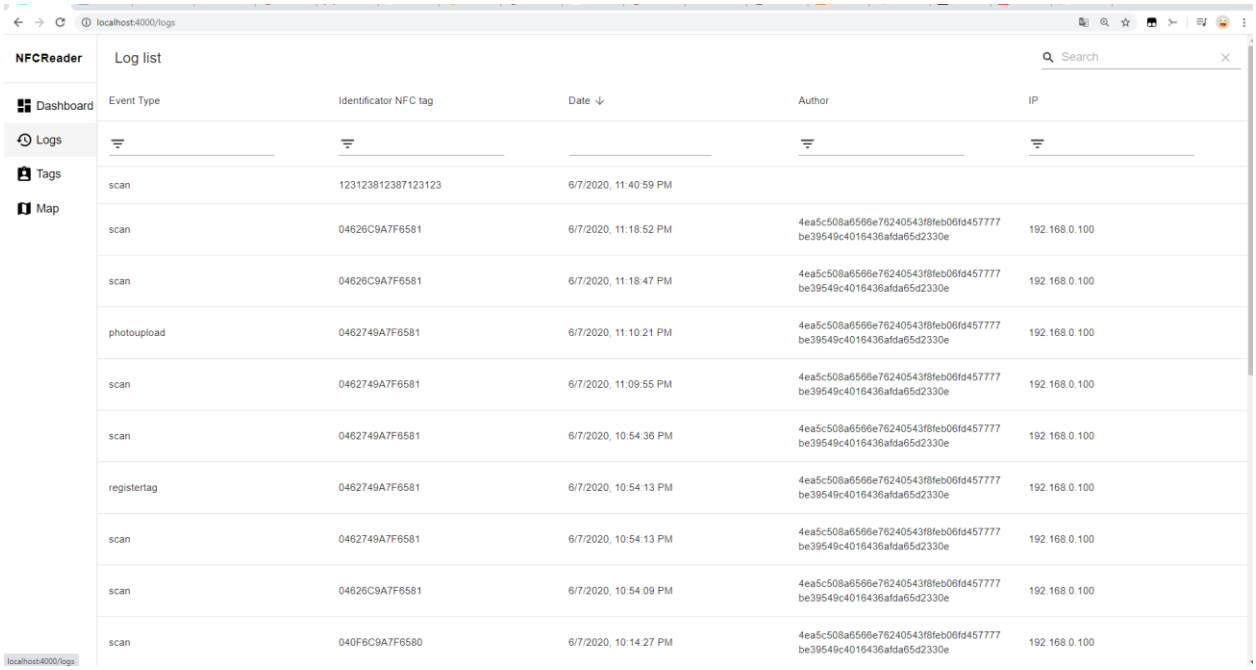


Рисунок 10

5.2.2 Сторінка з даними міток

Рисунок 11

На рисунку 11 зображено усі мітки, що є у базі даних. Вигляд міток відображено у стилі реальних карт-пропусків.

Кожна мітка відображає наступну інформацію:

- Ім’я
- Фотокартку

  Ідентифікатор
- Права доступу

Під кожною міткою розташовано кнопку «EDIT TAG», кнопка призначена для редагування даних первної мітки. Відображення інтерфейсу редагування є на рисунку 12.

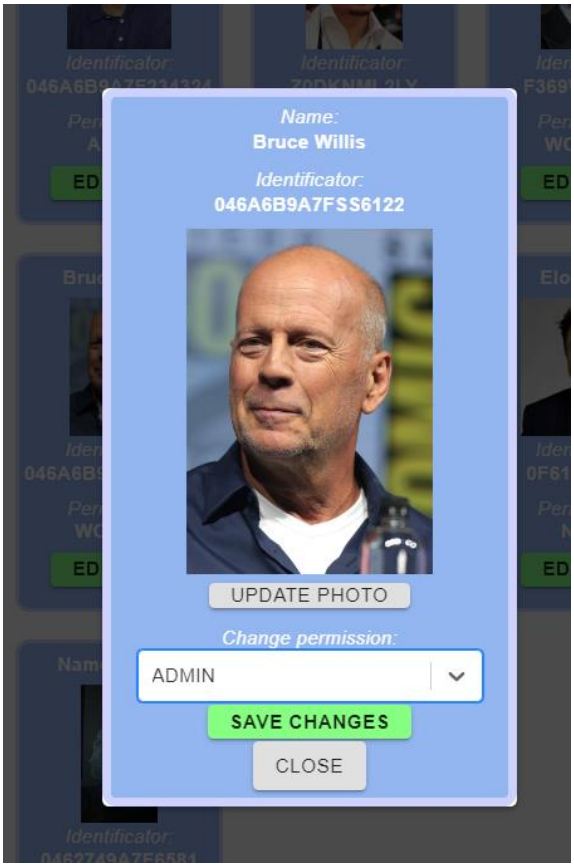


Рисунок 12

Інтерфейс має кнопку «upload photo» для завантаження фото з локальних файлів комп'ютеру адміністратора, має випадаюче меню з назвою «Change permission», пункти меню: Admin, Worker, None.

Після кліку по «SAVE CHANGES» до backend йде запит для оновлення даних цієї мітки.

5.2.3 Інтерактивна мапа

На рисунку 13 та 14 відображено інтерактивну карту. Карта має наступні елементи:

- Мітки, що розташовані за координатами сканувань
- Додаткову інформацію про мітку

Можливості:

- Переглядати розташування міток;
- Віддаляти та збільшувати масштаб мапи;
- Отримувати додаткову інформацію про мітки;

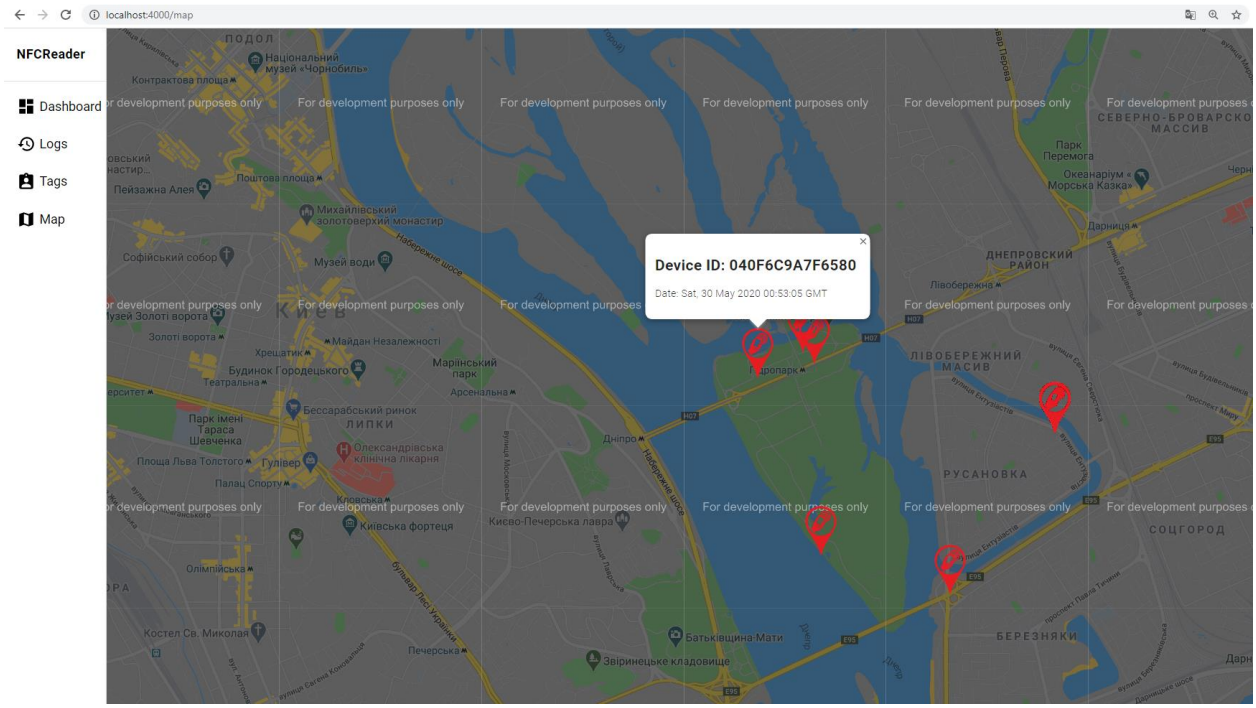


Рисунок 13

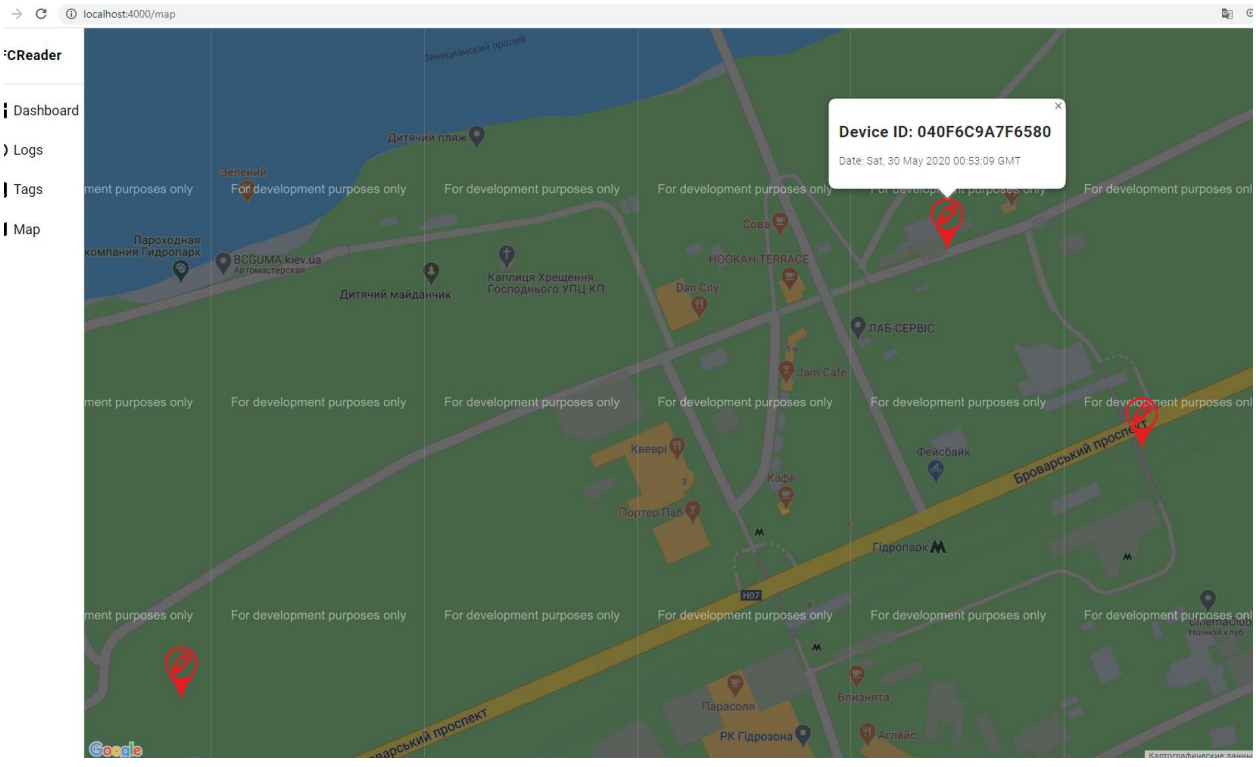


Рисунок 14

ВИСНОВКИ

У випускному кваліфікаційному проекті представлено результати практичних досліджень та аналізу теоретичних відомостей, що полягають у розробці системи контролю доступу для мобільної точки пропуску.

Під час розробки продукту було вивчено та застосовано клієнт-серверну архітектуру додатку, паттерн проектування MVC. При розробці серверної частини емпіричним способом було виявлено, що правильна побудова моделей, контроллерів забезпечить швидке та легке масштабування проекту.

Розробляючи серверну частину зробив висновок щодо неймінгу роутінгу – важливо створити і дотримуватися певну логіку найменувань у своєму додатку, це дозволить швидко орієнтуватися у великих проектах. Найкращим прикладом такого архітектурного стилю є REST API. Емпіричним шляхом також було з’ясовано, що треба правильно використовувати види HTTP запитів і не замінювати один іншим.

Під час виконання дипломної роботи я отримав досвід розробки додатків для різних платформ, що є цінним досвідом. Навчився розробляти френдлі-UX для різних платформ, а саме: для desktop WEB та Android.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1.Digipay [Електронний ресурс] : Стаття. All you need to know about NFC payments. Режим доступу : https://www.digipay.guru/blog/all-you-need-to-know-about-nfc-payments/

2. Number of smartphone users worldwide from 2016 to 2021 [Електронний ресурс] : Стаття. Statista. Режим доступу : https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

3. Mognoose. elegant mongodb object modeling for node.js [Електронний ресурс] : API Docs. Режим доступу : https://mongoosejs.com/docs/api.html

4. Features of React JS: Main Pros and Cons [Електронний ресурс] : Стаття. Режим доступу : https://acowebs.com/react-js-features/

5. Зображення моделі (рис.1) [Електронний ресурс] :Система безпеки компанії Intteks для бізнес центрів. Режим доступу : http://intteks.com.ua/typical-solutions/biznes-tsentry

6. A Simple HTTP/HTTPS Proxy in Node Js [Електронний ресурс] : Стаття. Режим доступу: https://dev.to/nimit95/a-simple-http-https-proxy-in-node-js-3810

7. How to Build and Structure a Node.js MVC Application [Електронний ресурс] : Стаття. Режим доступу : https://www.sitepoint.com/node-js-mvc-application/

7. Cloud DataBase features [Електронний ресурс] : Стаття. Режим доступу : https://www.ibm.com/cloud/learn/what-is-cloud-database#:~:text=Data%20organization%20on%20the%20cloud&text=Key%20features%3A,and%20managed%20by%20a%20provider

8. Express JS [Електронний ресурс]: Documentation. Режим доступу :https://expressjs.com/

9. React JS [Електронний ресурс]: Documentation. Режим доступу : https://reactjs.org/docs/getting-started.html

10. React Native [Електронний ресурс]: Documentation . Режим доступу : https://reactnative.dev/docs/getting-started

Додаток 1

Проектування та розроблення WEB – додатків на платформі систем контролювання доступу “Intteks Acs”

Специфікація

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ61172_20Б

Аркушів 1

Київ – 2020

Позначення	Найменування	Примітки
Документація		
	Записка.docx	Пояснювальна записка
Компоненти		
	App.js	Виконуваний файл backend серверу
	App.js	Виконуваний файл frontend серверу
	App.apk	Виконуваний файл мобільного додатку

Додаток 2

Проектування та розроблення WEB – додатків на платформі систем контролювання доступу “Intteks Acs”

Специфікація

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ61172_20Б

Аркушів 9

Київ – 2020

Код мобільного додатку:

```
import React, {Component} from 'react';
navigator.geolocation = require('@react-native-community/geolocation');
import NfcManager, { NfcTech, nfcManager } from 'react-native-nfc-manager';
import axios from 'axios';
import { Pages } from 'react-native-pages';
import ImagePicker from 'react-native-image-picker';
import defaultPrefences from './controller/preferenceContoller';
import imgNoAvatar from './public/noavatar.png';
import {Bars} from 'react-native-loader';
let url = "http://192.168.0.104:3000";
const options = {
  title: 'Select Avatar',
  storageOptions: {
  skipBackup: true,
  path: 'images',
  },
};
class App extends Component {
  constructor(props){
    super(props);
    this.state = {
      log: "",
      text: "",
      tagInfo: "",
      isScanned: 'notloaded',
      name:"",
      photo:"",
      permission:"",
```

```
        status: '',
        editing: false,
        location: null,
        scanButtonStatus:"Press to scan"
    }
}
render() {
    return (
        <Pages startPage={1}>
            <View style={styles.fullScreen}>
                <Text style={{fontSize:25}}>Quality Assurance page</Text>
                <TextInput
                    style={styles.textInput}
                    onChangeText={this.onChangeText}
                    autoCompleteType="off"
                    autoCapitalize="none"
                    autoCorrect={false}
                    placeholderTextColor="#888888"
                    placeholder="Enter text here" />
                <TextInput/>
                <TouchableOpacity
                    style={styles.buttonWrite}
                    onPress={this.writeData}>
                    <Text style={styles.buttonText}>Write</Text>
                </TouchableOpacity>
            </View>

            {this.state.isScanned == 'loaded' ?
                <View  style={styles.fullScreen}>
```

```
                {this.state.photo ?
                  <Image
                    style={styles.avatarBox}
                    resizeMode="contain"
                    source={{uri: this.state.photo}}
                    />
                  :<View style={styles.avatarBox}>
                    <Image
                      style={{width: "60%", height: 200}}
                      resizeMode="contain"
                      source={imgNoAvatar}
                      />
                    <Button onPress={()=>{this.setupPhoto()}} title="Upload image"/>
                  </View >}

                <View style={styles.infoBox}>
                  <Text style={styles.titleText}>IS REGISTERED:</Text>
                  <View style={this.state.status =='notregistered' ? styles.valueBoxAlert : styles.valueBox}>
                    {this.state.status =="registered" ? <Text style={styles.valueText}>Registered</Text>
                      :<Text style={styles.valueText}>This visitor not registered in system</Text>
                    }
                  </View>
                  <Text style={styles.titleText}>NAME:</Text>
                  <View style={styles.valueBox}>
                    <Text style={styles.valueText}>{this.state.name}</Text>
```

```
                    </View>
                    <Text style={styles.titleText}>PERMISSIONS:</Text>
                    <View style={this.state.permission ? styles.valueBox :
styles.valueBoxAlert}>
                      {this.state.permission ? <Text
style={styles.valueText}>{this.state.permission}</Text>
                        :<Text style={styles.valueText}>None</Text>
                      }
                    </View>
                  </View>
                  <TouchableOpacity
                    style={styles.buttonSend}
                    onPress={this.readData}>
                    <Text
style={styles.buttonText}>{this.state.scanButtonStatus}</Text>
                  </TouchableOpacity>
                </View>
                : this.state.isScanned == "loading" ?
                <View  style={styles.fullScreen}>
                  <Text  style={{fontSize:20,color:"#81b4f0"}}>Retrieving  data
from server</Text>
                  <Bars size={40} color="#81b4f0" />
                </View>
                : this.state.isScanned == "notloaded" ?
                <View style={styles.fullScreen}>
                  <Text  style={{fontSize:40,  fontWeight:'bold',
color:'#217eeb'}}>NFCreader</Text>
                  <TouchableOpacity
                    style={styles.buttonSend}
```

```
                        onPress={this.readData}>
                        <Text
style={styles.buttonText}>{this.state.scanButtonStatus}</Text>
                    </TouchableOpacity>
                    <View style={styles.log}>
                        <Text>{this.state.log}</Text>
                    </View>
                </View>
                : <View></View>
            }
        </Pages>
          )
        }
    }
    export default App;
```

Код серверної частини:

Основний виконуючий файл app.js:

```
var createError = require('http-errors');
var express = require('express');
var path = require('path');
var cookieParser = require('cookie-parser');
var logger = require('morgan');
var bodyParser = require('body-parser');
const mongoose = require('mongoose');
var indexRouter = require('./routes/index');
var eventRouter = require('./routes/event');
var APIrouter = require('./routes/api');
var app = express();
```

```
// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'jade');
app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));
app.use('/uploads', express.static(process.cwd() + '/uploads'))
var cors = require('cors');
app.use(cors());
app.use('/', indexRouter);
app.use('/event', eventRouter);
app.use('/api', APIrouter);
var db = mongoose.connection;
mongoose.connect('mongodb+srv://admin:kHsIKrhsssbYGoGFXS7@cluster0-
zzaup.mongodb.net/diploma?retryWrites=true&w=majority',    {useNewUrlParser:
true, useUnifiedTopology: true});
mongoose.set('useFindAndModify', false);
db.on('error', console.error.bind(console, 'connection error:'));
db.once('open', function() {
  console.log('connected to db');
});
// catch 404 and forward to error handler
app.use(function(req, res, next) {
  next(createError(404));
});
// error handler
app.use(function(err, req, res, next) {
```

```
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};
  // render the error page
  res.status(err.status || 500);
  res.render('error');
});
module.exports = app;
```

Код панелі адміністратора:

Головний виконуючий файл App.js:

```
import React from 'react';
import { BrowserRouter, Switch, Route, Link , browserHistory} from "react-router-dom";
import { createBrowserHistory } from 'history';
import { makeStyles , withStyles } from '@material-ui/core/styles';
import Sidebar from './components/Sidebar';
import LogsComponent from './components/Logs';
import MapComponent from './components/Map';
import TagComponent from './components/Tags';
import DashboardCompoonent from './components/Dashboard';
export const history = createBrowserHistory()
function App() {
  const drawerWidth = 130;
  const useStyles = makeStyles(theme => ({
    workArea:{
      width: `calc(100% - ${drawerWidth}px)`,
      marginLeft: drawerWidth,
```

```
    height:'100%'
  }}));
  const classes = useStyles();
  return (
    <BrowserRouter history={history}>
      <Sidebar/>
      <div className={classes.workArea}>
        <Route exact path="/logs"><LogsComponent/></Route>
        <Route exact path="/tags"><TagComponent/></Route>
        <Route exact path="/map"><MapComponent/></Route>
        <Route exact path="/"><DashboardCompoonent/></Route>
      </div>
    </BrowserRouter>
  )
}
export default App;
```

Додаток 3

Проектування та розроблення WEB – додатків на платформі систем контролювання доступу “Intteks Acs”

Специфікація

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ61172_20Б

Аркушів 6

Київ – 2020

АНОТАЦІЯ

Розроблена надає можливість створювати мобільні точки пропуску, контролювати процеси контролю, аналізувати події контролю.

Розроблене програмне забезпечене використовує MERN stack.

ЗМІСТ

Загальні відомості

Відповідно до назви дипломної роботи, розроблений продукт є система контролю доступу для мобільного зчитувача карт-пропусків.

Адміністратор повинен мати комп’ютер для взаємодії з системою контролю. База даних є хмарною і для взаємодії з нею адміністатору потрібен лише комп’ютер та інтернет.

Для використання додатку потрібно охоронцю запустити мобільний додаток для прикладати до нього мітки пропуску.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблений програмний засіб покликаний винуванти: процес контролю доступу, аналітику, збереженняя історії подій роботи системи. Це було реалізовано за допомогою кількох функцій системи, а саме:

— Компонентного підходу під час розробки;

— Записування усіх змін у журнал подій;

— Редагування прав карт-пропусків;

— Створення аналітики та інтерактивної мапи;

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Загальний принцип роботи додатку такий:

Для охоронця:

1) Охоронець вмикає мобільний додаток
2) Сканує дані з карт-пропусків
3) В залежності від відповіді додатку дозволяє чи забороняє прохід володарю картки на закриту територію

Для адміністратора:

1) Зайти у веб-панель адміністратора
2) Змінити права для певного користувача
3) Аналізувати дані з інтерактивної мапи
4) Переглядати журнал подій

ВХІДНІ І ВИХІДНІ ДАНІ

Дані, які надходять до системи від мобільного зчитувача карт-пропусків:

— Ідентифікатор мітки;

— Інформація з мітки;

— Фотокартка;

— Права доступу;

— Подія у журналі подій;

Вихідними даними в мобільному додатку є:

— інформація володаря картки-пропуску;

— Права доступу;

— фотокартка володаря картки;